UNIVERSIDAD DE GRANADA

PROYECTO FIN DE GRADO

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS

Ciber-AsesorIA/ST

Herramienta basada en IA generativa para asesorar a empresas en la mejora de su ciberseguridad. Subsistema de medidas técnicas.

Autor
D. Florin Emanuel Todor Gliga

Tutor
Dr. D. Pedro García Teodoro

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2026

Ciber-AsesorIA/ST: Herramienta basada en IA generativa para asesorar a empresas en la mejora de su ciberseguridad. Subsistema de medidas técnicas.

Florin Emanuel Todor Gliga

Palabras clave: Cumplimiento normativo, Esquema Nacional de Seguridad (ENS), NIS2, DORA, Grafo de conocimiento, Neo4j, IA generativa, Principio de Máximos.

Resumen

Ciber-AsesorIA es un asesor basado en IA generativa ideada para ayudar a las empresas a comprender sus obligaciones regulatorias en ciberseguridad (DORA, la Directiva NIS2 y el Esquema Nacional de Seguridad, ENS) y a evaluar su cumplimiento técnico frente a controles concretos. La herramienta está formada por **dos módulos complementarios** que operan de forma integrada de extremo a extremo: el módulo relativo al sistema normativo **(SN)** y el módulo relativo al sistema de medidas técnicas **(ST)**.
El módulo SN, desarrollado por mi compañera de proyecto, es una base de conocimiento aumentada con recuperación sobre las fuentes normativas que, a partir del perfil de una empresa, clasifica a la organización bajo los marcos que le aplican y deriva el conjunto de controles legislativos que está obligada a cumplir. El módulo ST, que documenta esta memoria, es el motor de evaluación de cumplimiento del proyecto y toma esos controles aplicables como entrada. Una capa de integración entre servicios (el coordinador) articula ambos módulos, de modo que el alcance normativo que produce el SN alimenta la evaluación técnica que realiza el módulo ST.

El módulo ST aborda el problema de la *inferencia normativa*: a partir del perfil regulatorio de una organización y de un control legislativo concreto, determina de forma automática qué requisitos técnicos le son exigibles y evalúa en qué grado los cumple. Para ello se apoya en un grafo de conocimiento construido sobre Neo4j, sobre el que se modelan los controles y requisitos del ENS, sus relaciones de equivalencia con NIS2 y DORA, y la lógica del *Principio de Máximos*, la regla que selecciona, para cada control,

la exigencia más restrictiva entre los tres marcos. La determinación de los requisitos aplicables se resuelve de forma determinista mediante una consulta al grafo, mientras que un agente conversacional basado en un modelo de lenguaje traduce cada requisito a lenguaje de negocio, conduce la auditoría requisito a requisito y emite un veredicto razonado y trazable, acompañado de una puntuación determinista. El motor de inferencia se valida mediante baterías de invariantes derivadas de las propias especificaciones normativas, que comprueban que sus resultados satisfacen la norma y no la implementación que las calcula.

Ciber-AsesorIA/ST: A Generative-AI Tool for Advising Companies on Improving Their Cybersecurity. Technical Measures Subsystem.

Florin Emanuel Todor Gliga

Keywords: Regulatory compliance, Spanish National Security Framework (ENS), NIS2, DORA, Knowledge graph, Neo4j, Generative AI, Principle of Maxima.

Abstract

Ciber-AsesorIA is a generative AI-based advisor designed to help companies understand their cybersecurity regulatory obligations (DORA, the NIS2 Directive, and the Spanish National Security Framework, ENS) and assess their technical compliance against specific controls. The tool consists of **two complementary modules** that operate in an integrated, end-to-end manner: the normative system module **(SN)** and the technical measures system module **(ST)**.
The SN module, developed by my project partner, is a retrieval-augmented knowledge base over the regulatory sources that, based on a company's profile, classifies the organization under the applicable frameworks and derives the set of legislative controls it is required to comply with. The ST module, which is the focus of this dissertation, is the project's compliance evaluation engine and takes those applicable controls as input. An inter-service integration layer (the coordinator) orchestrates both modules, ensuring that the regulatory scope produced by the SN feeds into the technical evaluation performed by the ST module.

The ST module addresses the problem of *regulatory inference*: given an organization's regulatory profile and a specific legislative control, it automatically determines which technical requirements are applicable and evaluates to what degree they are met. To achieve this, it relies on a knowledge graph built on Neo4j, which models the ENS controls and requirements, their equivalence relationships with NIS2 and DORA, and the logic of the *Principle of Maxima*—the rule that selects the most restrictive requirement among the three frameworks for each control. The determination of applicable requirements is resolved deterministically through a graph query, while a language model-based conversational agent translates each requirement into business language, conducts the audit requirement by requirement, and issues a reasoned, traceable verdict, accompanied by a deterministic score. The inference engine is validated using batteries of invariants derived from the normative specifications themselves, ensuring that its results satisfy the regulation rather than the implementation that computes them.

Dr. D. **Pedro García Teodoro**, Profesor del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Ciber-AsesorIA/ST: Herramienta basada en IA generativa para asesorar a empresas en la mejora de su ciberseguridad. Subsistema de medidas técnicas***, ha sido realizado bajo su supervisión por **Florin Emanuel Todor Gliga**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 22 de Junio de 2026.

El tutor:

Dr. D. Pedro García Teodoro

Agradecimientos

Deseo expresar mi más sincero agradecimiento a todas las personas que han contribuido a la realización de este trabajo.

En primer lugar, a mi tutor, el Dr. D. Pedro García Teodoro, por su orientación, dedicación y valiosos consejos a lo largo de todo el desarrollo del proyecto.

A mi familia, y muy especialmente a mi hermana Alexandra, por su apoyo incondicional, el cual ha sido fundamental para hacer más llevadero este proceso.

Por último, a mi compañera Laura, por su colaboración en el desarrollo de la herramienta *Ciber-AsesorIA* y su integración conjunta con el Sistema Normativo (SN).

Índice general

Índice de figuras

Índice de cuadros

Índice de listados

Capítulo 1

Introducción

1.1. Contexto y motivación

La transformación digital de la economía ha convertido la ciberseguridad en un factor crítico de supervivencia para cualquier organización. El impacto económico de la ciberdelincuencia alcanza cifras enormes: según Cybersecurity Ventures, se preveía que el coste global del cibercrimen alcanzaría los 10,5 billones de dólares USD en 2025 [1]. De medirse esta cifra como una economía nacional, situaría a la actividad delictiva como la tercera mayor del mundo, únicamente por detrás de Estados Unidos y China [1]. Esta escala refleja un problema que ha trascendido la esfera técnica para convertirse en un riesgo estratégico, operativo y reputacional de primer orden.

El panorama de amenazas en la Unión Europea corrobora esta tendencia. Según la Agencia de la Unión Europea para la Ciberseguridad (ENISA), en su informe anual que cubre el período del 1 de julio de 2024 al 30 de junio de 2025, se analizaron 4.875 incidentes [8]. Los datos revelan que la Administración Pública fue el sector más atacado, con el 38,2 % de los incidentes registrados; las entidades clasificadas como esenciales bajo la Directiva NIS2 representaron el 53,7 % del total [8]. En cuanto a los vectores de ataque inicial, el *phishing* (incluyendo *vishing*, correos maliciosos y publicidad maliciosa) se mantuvo como el dominante con aproximadamente el 60 % de los casos, seguido de la explotación de vulnerabilidades con el 21,3 % [8]. Este patrón de amenaza, creciente, diversificado y cada vez más sofisticado, presiona a las organizaciones para demostrar no solo conformidad formal, sino capacidades técnicas efectivas de defensa.

Ante este panorama, la legislación europea y nacional ha establecido un marco normativo de obligado cumplimiento. A nivel de la Unión Europea, el Reglamento (UE) 2022/2554 (DORA) aborda la resiliencia operativa digital del sector financiero y sus proveedores críticos [16], mientras que la

Directiva (UE) 2022/2555 (NIS2) establece medidas para alcanzar un nivel común elevado de ciberseguridad, ampliando el alcance a sectores esenciales e importantes [17]. En España, el Real Decreto 311/2022 regula el Esquema Nacional de Seguridad (ENS), aplicable tanto al sector público como a empresas privadas que suministran servicios o tecnología a administraciones públicas [20].

En paralelo a esta regulación de la ciberseguridad, la propia inteligencia artificial ha pasado a ser objeto de regulación: el Reglamento Europeo de Inteligencia Artificial (RIA), en vigor desde 2024 [18], y el proyecto de ley orgánica español para el buen uso y la gobernanza de la IA, aprobado en 2026 [12]. Bajo ese marco queda también la herramienta descrita en esta memoria, cuestión que se retoma en el capítulo de Conclusiones.

La brecha operativa es evidente. Para una organización típica, en especial las pequeñas y medianas empresas sin departamento de cumplimiento especializado, es extraordinariamente complejo: (a) determinar qué normativa le aplica en función de su perfil (sector, tamaño, criticidad); (b) resolver los solapamientos y conflictos entre marcos distintos; (c) traducir artículos legislativos abstractos en controles técnicos concretos y auditables. Precisamente este vacío entre la obligación regulatoria y la evidencia técnica verificable es lo que motiva el desarrollo del módulo del Sistema Técnico (ST) de *CiberAsesorIA*: un motor de evaluación capaz de inferir, a partir del perfil de una empresa, los controles técnicos aplicables bajo el principio más restrictivo entre ENS, NIS2 y DORA, y de evaluar su cumplimiento mediante una conversación guiada en lenguaje de negocio.

1.2. Definición del problema

El problema central que aborda este trabajo puede enunciarse de forma precisa: **dado el perfil regulatorio de una organización y un control legislativo concreto, ¿cómo determinar de manera ágil, trazable y objetiva qué requisitos técnicos debe cumplir y si efectivamente los cumple?** De esta formulación general se desprenden cuatro subproblemas que el módulo del Sistema Técnico debe resolver de forma encadenada.

1. Inferencia de aplicabilidad

Un mismo artículo legislativo no se traduce en el mismo conjunto de controles técnicos para todas las empresas. El control aplicable depende del perfil: la categoría ENS (BÁSICA, MEDIA o ALTA), el tipo de entidad bajo NIS2 (Esencial, Importante o ninguna) y la sujeción o no a DORA. El sistema debe, por tanto, partir de un texto normativo no estructurado y

localizar entre un catálogo amplio de controles técnicos aquellos semánticamente pertinentes, descartando los falsos positivos: controles que comparten vocabulario de seguridad pero evalúan una práctica distinta.

2. Resolución del solapamiento normativo (Principio de Máximos)

Una empresa puede estar sujeta simultáneamente a ENS, NIS2 y DORA, y cada marco puede exigir un nivel de rigor diferente sobre el mismo control. El problema no se resuelve sumando requisitos, sino determinando, para cada control, el requisito más restrictivo aplicable entre los tres orígenes regulatorios. Esta lógica, el "Principio de Máximos", debe además contemplar las exclusiones y refuerzos específicos que cada normativa impone según el tipo de entidad, lo que la convierte en una operación que no puede reducirse a una simple tabla de correspondencias estática.

3. Traducción de lo técnico a lo comprensible

El destinatario de la evaluación no es un auditor experto, sino un responsable de negocio. Un requisito técnico expresado en su forma normativa cruda resulta ininteligible o, peor aún, induce a error: una empresa puede creer que cumple algo porque "lo hace de manera informal", cuando el requisito exige evidencia documental. El sistema debe reformular cada requisito en lenguaje de negocio, distinguiendo explícitamente entre lo que debe estar documentado (evidencia) y lo que basta con demostrar que se practica (verificación).

4. Emisión de un veredicto objetivo y trazable

El resultado de la evaluación no puede depender de la subjetividad del momento ni del fraseo de una respuesta. El sistema debe agregar los veredictos individuales aplicando reglas deterministas (en particular, el incumplimiento de un requisito marcado como esencial invalida el control completo) y dejar constancia auditable de qué se evaluó, con qué resultado y por qué.

Las aproximaciones tradicionales abordan este problema de forma insuficiente. La consultoría manual es precisa pero costosa, lenta y difícilmente escalable a pymes. Las herramientas de cumplimiento existentes tienden a apoyarse en cuestionarios estáticos y listas de verificación predefinidas: capturan el estado declarado del cumplimiento, pero no infieren dinámicamente el control aplicable a partir del texto normativo ni resuelven el solapamiento entre marcos (un análisis detallado de estas soluciones se desarrolla en el capítulo de Estado del Arte). El problema no es de gestión documental del cumplimiento, sino de **inferencia normativa**: razonar desde el perfil

de la empresa y el texto legislativo hasta el requisito técnico exacto y su verificación.

1.3. *Ciber-AsesorIA*: visión global del proyecto conjunto

Ciber-AsesorIA es un sistema de asesoramiento basado en IA generativa, desarrollado como proyecto conjunto, cuyo objetivo es guiar a una empresa desde la pregunta inicial ("¿qué normativa de ciberseguridad me obliga?") hasta un informe de cumplimiento técnico sobre controles concretos. Arquitectónicamente, se compone de una interfaz web y tres servicios de *backend* coordinados (véase la Figura 1.1):

- ***Frontend* (Next.js).** Recoge el perfil de la empresa mediante un asistente por pasos, realiza una clasificación regulatoria preliminar y ofrece la interfaz de chat con la que el usuario conversa durante la evaluación.
- **Sistema Normativo (SN).** Determina qué controles legislativos son aplicables al perfil y elabora el informe narrativo final. Se trata del subsistema en conjunto.
- **Sistema Técnico (ST).** Motor de evaluación técnica, objeto de esta memoria. Recibe cada control legislativo, infiere los requisitos técnicos aplicables aplicando el Principio de Máximos y evalúa su cumplimiento mediante conversación.
- **Coordinador.** Orquestador central que recibe el perfil del *frontend*, solicita los controles al SN, gestiona una conversación con el módulo del ST por cada control y recopila los veredictos resultantes para trasladarlos al SN y devolver el resultado final.

El flujo de extremo a extremo, que se detalla más adelante y se ilustra como diagrama de secuencia en el Anexo D, es en resumen el siguiente: el *frontend* envía el perfil al Coordinador; este obtiene del SN la lista de controles aplicables y, para cada uno, orquesta una evaluación con el módulo del ST; finalmente, los resultados se agregan y se devuelven al usuario a través del *frontend*.

Reparto de responsabilidades

El proyecto fue desarrollado por dos personas. El Sistema Normativo y la mayor parte del *frontend* fueron responsabilidad de la otra integrante del

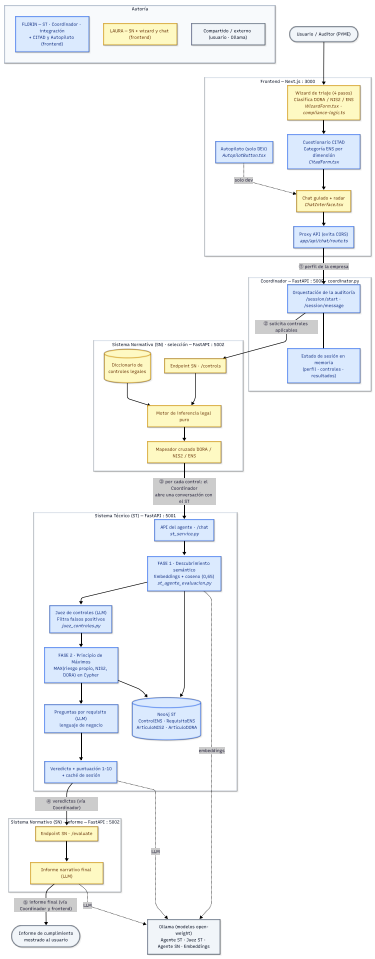

Figura 1.1: Arquitectura funcional de *Ciber-AsesorIA*: los tres servicios de *backend* (Sistema Normativo, Sistema Técnico y Coordinador) y el *frontend*, con la comunicación entre módulos y el reparto de autoría. El código de color distingue el trabajo del autor de esta memoria (azul) del de la otra integrante del equipo (amarillo).

equipo; este reparto se refleja mediante un código de color en la Figura 1.1. El trabajo del autor de esta memoria comprende el **módulo del Sistema Técnico (ST) en su totalidad**: el motor de inferencia normativa (búsqueda semántica de controles, filtrado mediante un juez basado en LLM y resolución del Principio de Máximos), el agente de evaluación conversacional que dialoga con el usuario y emite el veredicto técnico, y el modelo de datos en grafo que sustenta la base del sistema ST. Comprende, además, **toda la capa de comunicación entre servicios vía API**: la integración ST–Coordinador–SN y la comunicación con el *frontend*. A ello se suman dos piezas concretas del *frontend*, de autoría propia.

La primera es el **componente de clasificación CITAD** (Confidencialidad, Integridad, Trazabilidad, Autenticidad y Disponibilidad), el cuestionario de impacto mediante el cual el usuario determina la categoría de seguridad ENS de su sistema, la entrada que da comienzo a la evaluación técnica. La segunda es el **agente de Autopiloto**, una herramienta interna de desarrollo (visible únicamente en modo `dev`, nunca en producción) que simula las respuestas de un usuario de PYME mediante un modelo de lenguaje ligero y un conjunto de perfiles de madurez, y permite recorrer de forma automática el flujo de evaluación completo durante las pruebas. En suma, el módulo del ST no es solo el motor de evaluación: es también lo que une los tres subsistemas y los convierte en un producto funcional de extremo a extremo.

Alcance de esta memoria

Acorde con ese reparto, esta memoria se centra en el módulo del Sistema Técnico y en la capa de integración. La clasificación regulatoria previa (decidir bajo qué marcos cae una empresa) y el resto del *frontend* se describen únicamente en la medida necesaria para contextualizar el módulo del ST, ya que constituyen el trabajo de la otra integrante del equipo; se exceptúan el componente de clasificación CITAD y el agente de Autopiloto, de autoría propia, que sí se documentan como parte del trabajo descrito en esta memoria. La arquitectura interna del módulo del ST, su modelo de datos y los contratos de comunicación entre servicios se detallan en los capítulos correspondientes.

1.4. Organización de la memoria

El resto de la memoria se estructura en los siguientes capítulos:

- **Capítulo 2. Análisis de requisitos y funcionalidad de la herramienta:** objetivos del trabajo, funcionalidad y flujo de uso, requisitos

funcionales y no funcionales, tareas a desarrollar, recursos, metodología, planificación temporal y presupuesto.

- **Capítulo 3. Estado del arte y marco regulatorio:** revisión de las herramientas de cumplimiento existentes y de los enfoques basados en grafos de conocimiento, caracterización de los marcos normativos (DORA, NIS2 y ENS) y del reto de su interoperabilidad.
- **Capítulo 4. Análisis funcional y diseño del módulo del Sistema Técnico (ST):** flujo de evaluación, las dos fases del motor de inferencia, el modelo de datos, los requisitos no funcionales y el modelado del grafo de conocimiento sobre el que se resuelve el Principio de Máximos.
- **Capítulo 5. Implementación:** desarrollo del entorno, del agente de evaluación y de la capa de integración entre servicios.
- **Capítulo 6. Pruebas y validación:** validación del motor de inferencia y análisis de los resultados.
- **Capítulo 7. Conclusiones y trabajo futuro:** principales aportaciones, consideraciones legales y éticas, limitaciones y líneas de trabajo futuro.

Capítulo 2

Análisis de requisitos y funcionalidad de la herramienta

En este capítulo se definen los objetivos y la funcionalidad del módulo del Sistema Técnico, sentando las bases para su posterior diseño. A lo largo del texto se desgranan los requisitos técnicos y operativos de la herramienta, y se detalla el proceso de planificación llevado a cabo para su desarrollo: desde la distribución de tareas y recursos hasta el cronograma de trabajo y la estimación presupuestaria.

2.1. Objetivos

2.1.1. Objetivo general

El objetivo general de este trabajo es **diseñar e implementar el módulo del Sistema Técnico (ST)** de *Ciber-AsesorIA*: el motor de evaluación técnica de cumplimiento que, **apoyándose en un grafo de conocimiento normativo**, infiere automáticamente los requisitos técnicos aplicables a una empresa para un control legislativo dado, evalúa su cumplimiento emitiendo un **veredicto razonado y trazable**, y **se integra con el resto del sistema** (Coordinador, SN y *frontend*).

2.1.2. Objetivos específicos

Del objetivo general se derivan los siguientes objetivos específicos:

- **OE1. Construir el grafo de conocimiento normativo en Neo4j**:

estructurar los controles y requisitos técnicos del ENS a partir de fuentes regulatorias no estructuradas, vectorizarlos mediante *embeddings* y modelar sus relaciones de equivalencia con NIS2 y DORA a partir de los mapeos de correspondencia entre marcos. Este grafo es la base de datos sobre la que se sustenta el resto del sistema.

- **OE2. Inferir los controles técnicos relevantes** para un control legislativo dado, mediante búsqueda semántica vectorial sobre el grafo de conocimiento (*embeddings* + Neo4j).
- **OE3. Filtrar los falsos positivos** de esa búsqueda mediante un juez basado en un modelo de lenguaje (LLM), que descarte los controles semánticamente próximos pero no pertinentes.
- **OE4. Resolver el requisito más restrictivo entre normativas**, aplicando el Principio de Máximos sobre ENS, NIS2 y DORA directamente en la consulta al grafo (*Cypher*).
- **OE5. Conducir una auditoría conversacional en lenguaje de negocio**, evaluando requisito a requisito y traduciendo cada exigencia técnica a un lenguaje comprensible para un responsable no experto.
- **OE6. Emitir un veredicto de cumplimiento trazable y una puntuación determinista (escala 1–10)**: agregar los veredictos por control mediante reglas deterministas (en particular, la regla del requisito esencial) y acompañar el resultado de una traza de auditoría que documente qué se evaluó, con qué resultado y por qué.
- **OE7. Clasificar la categoría ENS de la empresa** (BÁSICA, MEDIA o ALTA) a partir del impacto en las cinco dimensiones de seguridad (CITAD).
- **OE8. Integrar el módulo del ST con el resto del sistema vía API**, definiendo los contratos de comunicación con el Coordinador (y, a través de él, con el SN y el *frontend*) y la gestión de sesión multiturno.
- **OE9. Validar sistemáticamente el motor de inferencia** mediante baterías de invariantes derivadas de las especificaciones normativas de origen (las reglas del PCE-NIS2, el *crosswalk* ENS–DORA y la tabla de aplicabilidad de la guía CCN-STIC-892), de modo que las pruebas comprueben que el motor satisface la norma y no que reproduce su propio comportamiento.

2.2. Funcionalidad y flujo de uso de la herramienta

La meta de *Ciber-AsesorIA* es **acercar la auditoría de ciberseguridad a un responsable de negocio** sin exigirle conocimientos técnicos ni legales avanzados. Esta premisa condiciona toda la funcionalidad del módulo del Sistema Técnico: la interacción es conversacional, en lenguaje de negocio, y el sistema nunca pide al usuario que interprete por sí mismo un artículo legislativo o un control técnico.

El módulo del ST no funciona de forma aislada, sino como uno de los servicios de la arquitectura distribuida descrita en la Sección 1.3. El recorrido de extremo a extremo (*end-to-end*) en el que participa se organiza en cuatro fases:

1. **Triaje y perfilado.** El usuario completa en el *frontend* un cuestionario progresivo del que se obtiene el perfil de la empresa (sector, tamaño, vinculación con el sector público y, mediante el componente CITAD, el nivel de impacto en cada dimensión de seguridad). El Coordinador remite ese perfil al Sistema Normativo, que dictamina la lista de **controles legislativos** aplicables.

2. **Preparación e inferencia (entrada del módulo del ST).** El Coordinador abre una conversación con el módulo del ST **por cada control legislativo**. El módulo del ST recibe una unidad de trabajo bien delimitada, el control y el perfil de la empresa, y cruza ese requisito legal genérico con su modelo técnico para resolver **qué requisitos técnicos concretos** debe auditar y con qué nivel de exigencia.

3. **Auditoría conversacional.** El módulo del ST recorre los requisitos uno a uno, formulando preguntas en lenguaje de negocio y recogiendo las respuestas del usuario a través del Coordinador y el *frontend*. Al completar cada control emite un **veredicto parcial** (cumple / no cumple), una puntuación de 1 a 10 y una traza de auditoría.

4. **Emisión del veredicto final.** Agotados todos los controles técnicos, el Coordinador agrega los veredictos y los devuelve al Sistema Normativo, que interpreta los resultados técnicos y elabora el informe de cumplimiento final. La frontera de responsabilidades es nítida: el módulo del ST produce el veredicto de *cada control legislativo* a través de la evaluación de los *controles técnicos*, mientras que el **informe global** corresponde al SN.

El detalle de las dos fases internas del motor (descubrimiento semántico de controles y resolución del Principio de Máximos), del modelo de datos que circula por el sistema y de la traza de auditoría se aborda en el Capítulo 4, dedicado al análisis funcional y al diseño.

2.3. Requisitos funcionales

Los requisitos funcionales delimitan las acciones concretas y verificables que el módulo del Sistema Técnico debe ser capaz de ejecutar. Se han identificado los siguientes:

- **RF-01. Recepción de la unidad de trabajo.** El sistema debe aceptar, a través de su API, un control legislativo (identificador, título y descripción) junto con el perfil regulatorio de la empresa, y tratarlo como una unidad de evaluación independiente.
- **RF-02. Descubrimiento semántico de controles técnicos.** El sistema debe localizar, entre todo el catálogo de controles técnicos del ENS, aquellos semánticamente pertinentes para el control legislativo recibido, mediante búsqueda vectorial sobre el grafo de conocimiento y un umbral de similitud.
- **RF-03. Filtrado de falsos positivos.** El sistema debe depurar los candidatos anteriores con un juez basado en un LLM, descartando los controles próximos en vocabulario pero no pertinentes para auditar el artículo.
- **RF-04. Inferencia del requisito más restrictivo.** El sistema debe determinar, para cada control, los requisitos técnicos exigibles y su nivel, resolviendo de forma autónoma la confluencia de ENS, NIS2 y DORA mediante el Principio de Máximos, con sus refuerzos y exclusiones.
- **RF-05. Auditoría conversacional en lenguaje de negocio.** El sistema debe evaluar requisito a requisito, formulando preguntas comprensibles para un usuario no experto y distinguiendo entre exigencias de evidencia documental y de verificación de la práctica.
- **RF-06. Emisión de un veredicto trazable y puntuación determinista.** El sistema debe agregar los veredictos por control mediante reglas deterministas (incluida la regla del requisito esencial), producir una puntuación en escala 1–10 y reutilizar veredictos ya emitidos mediante una caché de sesión.
- **RF-07. Clasificación CITAD de la categoría ENS.** El sistema debe derivar la categoría de seguridad ENS (BÁSICA, MEDIA o ALTA) a partir del impacto declarado por el usuario en las cinco dimensiones de seguridad.
- **RF-08. Generación de la traza de auditoría.** El sistema debe registrar, por sesión, qué controles técnicos se evaluaron, su veredicto

y razón, la similitud que motivó su selección y el veredicto final, de forma que la evaluación sea reconstruible a posteriori.

- **RF-09. Interoperabilidad estructurada.** El módulo debe comunicarse, mediante contratos de interfaz definidos, con el Coordinador (y a través de él con el SN y el *frontend*) y con los modelos de lenguaje que articulan la fase conversacional, gestionando una sesión multi-turno.

2.4. Requisitos no funcionales

Más allá de su funcionalidad, el sistema debe satisfacer una serie de propiedades transversales que condicionan toda la solución:

- **RNF-01. Determinismo de la evaluación.** El motor de inferencia (el Principio de Máximos) y la puntuación final (1–10) deben ser funciones puras del perfil y del resultado de cada requisito, sin intervención de azar. Las partes apoyadas en LLM se ejecutan a **temperatura cero** para maximizar la reproducibilidad.
- **RNF-02. Trazabilidad y auditabilidad.** El resultado no puede ser una caja negra: cada requisito evaluado debe conservar sus orígenes normativos y el sistema debe generar una traza de auditoría que justifique qué se evaluó, con qué resultado y por qué.
- **RNF-03. Rendimiento y resiliencia.** La evaluación debe acomodar la latencia de los modelos de lenguaje mediante tiempos de espera amplios y una degradación elegante, de modo que un fallo del modelo nunca bloquee la evaluación.
- **RNF-04. Extensibilidad y escalabilidad.** El ecosistema regulatorio es volátil; el diseño basado en grafo debe permitir añadir o modificar normativas, controles y mapeos sin reescribir el núcleo de inferencia.
- **RNF-05. Privacidad y soberanía del dato.** La inferencia debe poder ejecutarse **en local** (modelos de pesos abiertos sobre infraestructura propia), evitando depender de APIs de terceros y manteniendo los datos sensibles de la empresa dentro de un entorno controlado.

El detalle de los mecanismos con los que el sistema satisface estas propiedades (la pureza del cálculo, la traza de auditoría y la política de *timeouts* y degradación) se documenta en el Capítulo 4.

2.5. Tareas a realizar

El desarrollo del módulo del Sistema Técnico se descompone en el siguiente conjunto de tareas de ingeniería, que se corresponden con los objetivos específicos enunciados en la Sección 2.1:

- **T1. Análisis y extracción del conocimiento normativo técnico.** Identificación de los controles y requisitos técnicos del ENS, de su nivel de exigencia y de la marca de requisito esencial (guía CCN-STIC-808), así como de los mapeos de correspondencia ENS–NIS2 y ENS–DORA.
- **T2. Construcción del grafo de conocimiento en Neo4j.** Modelado de la ontología, ingesta de los datos estructurados y vectorización de los controles mediante *embeddings* (OE1).
- **T3. Motor de descubrimiento semántico (Fase 1).** Implementación de la búsqueda vectorial con umbral de similitud y del juez basado en LLM para el filtrado de falsos positivos (OE2, OE3).
- **T4. Motor de inferencia normativa (Fase 2).** Traslado del Principio de Máximos sobre ENS, NIS2 y DORA a una única consulta *Cypher* sobre el grafo, con refuerzos y exclusiones (OE4).
- **T5. Agente conversacional de auditoría.** Evaluación requisito a requisito en lenguaje de negocio, juicio de cumplimiento, agregación determinista y puntuación 1–10 con traza de auditoría (OE5, OE6).
- **T6. Componente de clasificación CITAD.** Cuestionario de impacto por dimensiones que determina la categoría de seguridad ENS de la empresa (OE7).
- **T7. Coordinador e integración por API.** Orquestación del estado de sesión y de las llamadas HTTP entre *frontend*, SN y ST, y definición de los contratos de comunicación (OE8).
- **T8. Infraestructura de IA local.** Despliegue y configuración de los modelos de lenguaje y de *embeddings* sobre Ollama, incluido el acceso al servidor de inferencia de la Universidad de Granada.
- **T9. Pruebas y validación.** Baterías de invariantes derivadas de las especificaciones normativas (PCE-NIS2, *crosswalk* ENS–DORA, CCN-STIC-892) y pruebas de regresión e integración (OE9).
- **T10. Elaboración de la memoria técnica.** Documentación continua del estado del arte, las decisiones de diseño, la implementación, los resultados y las conclusiones del proyecto.

2.6. Recursos hardware, software y humanos

El módulo del Sistema Técnico se construyó con software de código abierto, modelos de pesos abiertos y servicios en la nube con capa gratuita, por lo que su coste material fue mínimo.

2.6.1. Recursos humanos

Ciber-AsesorIA es un Trabajo Fin de Grado conjunto. El **módulo del Sistema Técnico y la capa de integración** corresponden a este trabajo, mientras que el Sistema Normativo y el *frontend* fueron responsabilidad de la otra integrante del equipo; ambos miembros acordamos de forma conjunta los principios de diseño, el reparto de tareas y los contratos de la API. Las personas implicadas en el proyecto son:

- **Autor del PFG (Sistema Técnico e integración):** análisis normativo técnico, grafo de conocimiento, motor de inferencia, agente conversacional, Coordinador y validación (estimación de 300 h de dedicación directa).
- **Compañera del PFG conjunto (Sistema Normativo y *frontend*):** diseño e implementación de la interfaz de usuario interactiva, desarrollo del Sistema Normativo y generación del informe global de cumplimiento (estimación de 300 h de dedicación directa, integradas a través de los contratos de interfaz consensuados).
- **Tutor académico:** supervisión metodológica, seguimiento periódico y validación del trabajo.
- **Soporte técnico de la Universidad de Granada:** apoyo puntual en el acceso al servidor de inferencia compartido.

2.6.2. Recursos hardware

- **Equipo de desarrollo personal:** un ordenador portátil de gama media-alta, suficiente para ejecutar el entorno de desarrollo, los servicios *backend* (FastAPI) y el *frontend* (Node.js / Next.js) en local.
- **Servidor de inferencia (GPU) de la Universidad de Granada:** accedido mediante túnel SSH, ejecuta sobre `Ollama` los modelos de lenguaje (`qwen2.5` de 14B para el agente y 32B para el juez) y el modelo de *embeddings* (`mxbai-embed-large`). Su uso, sin coste para el proyecto, agilizó notablemente la velocidad del motor de razonamiento.

- **Instancia Neo4j Aura (nube):** base de datos de grafo gestionada, en su capa gratuita, donde reside el grafo de conocimiento técnico del módulo del ST.

2.6.3. Recursos software

- **Lenguajes y entorno:** Python (servicios *backend*, motor de inferencia y pruebas) y TypeScript (componente CITAD del *frontend*), sobre el IDE Visual Studio Code.
- **Librerías y *frameworks*:** `FastAPI` para los servicios, el controlador oficial de `Neo4j` y *Cypher* para el grafo, `Ollama` como entorno de inferencia local, y Next.js/React para la integración con el *frontend*.
- **Modelos:** `qwen2.5` (14B y 32B) y `mxbai-embed-large`, todos de **pesos abiertos** y sin coste de licencia.
- **Control de versiones y documentación:** un único repositorio centralizado en `GitHub`, `Google Workspace` para la documentación compartida, y LaTeX / Overleaf para la redacción de la memoria.

2.7. Metodología de desarrollo y coordinación del equipo

Para el desarrollo de este sistema distribuido y abordado por dos personas con responsabilidades diferenciadas se adoptó una **metodología ágil**, integrando principios de Scrum. Dado que tanto los requisitos legales como las capacidades de los modelos de lenguaje cambian con rapidez, se optó por un enfoque iterativo e incremental: el trabajo se estructuró en **iteraciones (*sprints*) de dos o tres semanas** que coincidían con las reuniones de seguimiento con el tutor, desde septiembre de 2025 hasta la fase final de redacción. Cada *sprint* comenzaba con la fijación de objetivos concretos y terminaba con un incremento funcional del software o un avance de investigación (por ejemplo, la preparación del grafo de conocimiento, la puesta en producción del motor de inferencia o la integración del Coordinador).

Coordinación del equipo

Antes de abordar el desarrollo individual, fue crucial acordar el diseño general de la herramienta, las responsabilidades de cada miembro, la definición de las interfaces y la estructura de los subsistemas. El reparto se fijó de la siguiente manera:

- **Sistema Normativo y *frontend* (compañera):** interfaz gráfica, motor de inferencia legal que dictamina qué marcos y artículos aplican, y generación del informe de cumplimiento final.
- **Sistema Técnico e integración (autor):** grafo de conocimiento en Neo4j, infraestructura de IA local con modelos abiertos sobre Ollama, agente conversacional de auditoría técnica y Coordinador encargado del estado de sesión y del arbitraje de las llamadas HTTP entre *frontend*, SN y ST.

Esta delimitación precisa minimizó los conflictos en el código y permitió que ambos avanzáramos en paralelo, interactuando únicamente a través de contratos previamente documentados y consensuados. Como herramientas de seguimiento se emplearon `Git`/`GitHub` (**repositorio privado**), `Google Drive` para la documentación no relacionada con el código y comunicación continua para la resolución de dudas y el debate de arquitectura.

2.8. Planificación temporal

El esfuerzo de desarrollo se enmarca en el curso académico 2025–2026. El proyecto arrancó con una **fase previa de definición**, en la que la idea y el alcance se perfilaron con el tutor desde agosto de 2025 hasta septiembre, y el desarrollo propiamente dicho se planificó desde septiembre de 2025 hasta la defensa en junio de 2026. Siguiendo el enfoque iterativo adoptado, las tareas descritas en la Sección 2.5 se secuenciaron como muestra el diagrama de *Gantt* de la Figura 2.1, que refleja además las pausas por exámenes y la doble fase de documentación: la redacción continua a lo largo del curso y el volcado final a LaTeX (Overleaf) ya en junio.

2.9. Presupuesto tentativo del proyecto

Aunque este Trabajo Fin de Grado es de fines académicos, resulta un ejercicio de ingeniería indispensable estimar el coste económico que supondría abordar un proyecto de esta magnitud en el mercado laboral real. Dado que *Ciber-AsesorIA* es un producto conjunto, el presupuesto que sigue corresponde al **producto resultante completo**, si bien la narrativa subraya las decisiones propias del módulo del Sistema Técnico.

De acuerdo con estimaciones del sector informático español, la tarifa de un desarrollador o consultor de software junior o *mid-level* se sitúa entre 18 y 30 € por hora facturable; para este cálculo se adopta una tarifa conservadora de 20 €/h. Considerando un equipo de **2 desarrolladores** con una carga

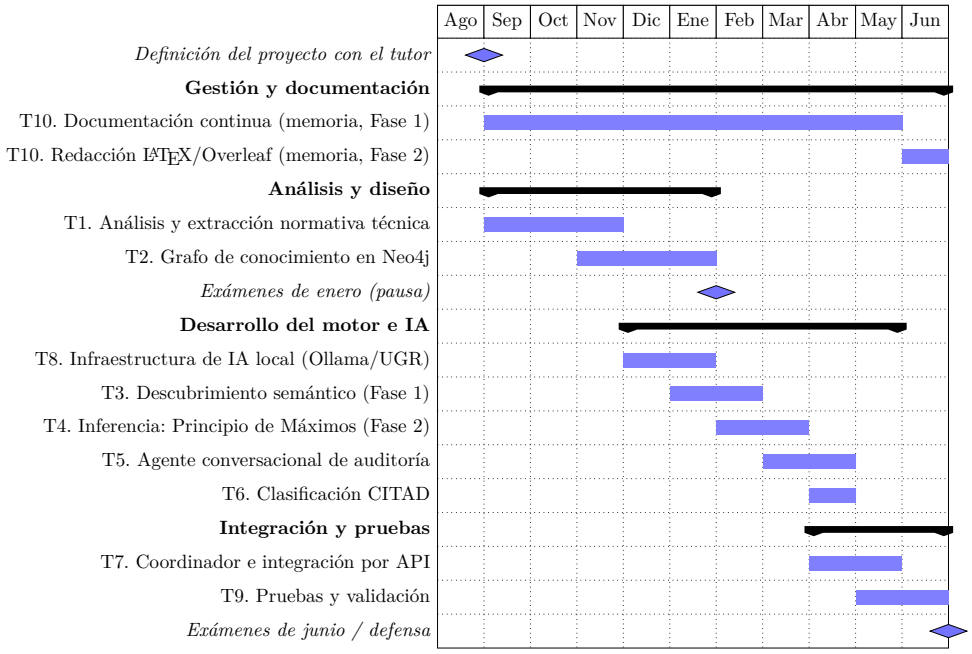

Figura 2.1: Diagrama de *Gantt* de la planificación temporal del desarrollo del módulo del Sistema Técnico (agosto de 2025 – junio de 2026): una fase previa de definición con el tutor en agosto, el análisis y el modelado del grafo durante el primer cuatrimestre, el grueso del desarrollo del motor tras la pausa de exámenes de enero, y la integración, las pruebas y el volcado final de la memoria a LaTeX en el tramo final.

de **300 horas** de dedicación directa cada uno, acorde a las horas estimadas para una asignatura de 12 créditos, el coste salarial asciende a 600 horas conjuntas.

En cuanto a la infraestructura material, se incluye la amortización de 2 portátiles personales (1.200 € cada uno), aplicando una vida útil de 3 años (33 % anual) y la fracción proporcional a los nueve meses de ejecución (0,75 años). De forma crítica, el presupuesto integra el uso de **infraestructura de inferencia local**: la decisión de optar por una arquitectura *on-premise* en lugar de depender de APIs de terceros responde a una necesidad estratégica y regulatoria, garantizar la privacidad en el tratamiento de los datos de las empresas, manteniéndolos dentro del entorno controlado de la organización y facilitando el cumplimiento del RGPD. Finalmente, se incorporan unos costes de despliegue *low-cost* y los costes administrativos del Régimen Especial de Trabajadores Autónomos (RETA) bajo la tarifa plana [24]. Las licencias de software son nulas al apoyarse el proyecto íntegramente en tecnologías de código abierto y modelos de pesos abiertos.

Este presupuesto sitúa el coste de producción del proyecto completo

Cuadro 2.1: Presupuesto estimado del producto *Ciber-AsesorIA* (sin IVA).

Concepto	**Cálculo estimado**	**Coste**
Personal IT (2 desarrolladores)	2 pers. × 300 h × 20,00 €/h	12.000,00 €
Amortización de hardware (2 portátiles)	(1.200 € × 33,3 %) × 0,75 × 2	600,00 €
Infraestructura de inferencia local	Uso / amortización de servidor	450,00 €
Costes de despliegue (*low-cost*)	Contenedores y entorno local	300,00 €
Costes administrativos (autónomos)	88,64 €/mes × 2 × 9 meses	1.595,52 €
Licencias y software	Código abierto / pesos abiertos	0,00 €
Subtotal de desarrollo		14.945,52 €
Gastos generales e imprevistos (10 %)	10 % s/ subtotal	1.494,55 €
Presupuesto total (base imponible)		**16.440,07 €**

en **16.440,07 euros**, reflejando un escenario de desarrollo profesional *on-premise* y contenido bajo las condiciones económicas del mercado actual.

Capítulo 3

Estado del arte y marco regulatorio

3.1. Herramientas actuales de auditoría y asesoramiento

Antes de situar la aportación de este trabajo conviene revisar cómo aborda el mercado el problema del cumplimiento normativo. El panorama puede ordenarse en dos grandes familias, las plataformas de Gobernanza, Riesgo y Cumplimiento (GRC) y los enfoques basados en grafos de conocimiento, a partir de cuya comparación se delimita, al final de la sección, el hueco que ocupa el módulo del Sistema Técnico.

3.1.1. Plataformas GRC y *suites* de cumplimiento

Las plataformas de Gobernanza, Riesgo y Cumplimiento (GRC) constituyen la respuesta comercial dominante a la gestión del cumplimiento. Su propósito es centralizar y automatizar las tareas de gobierno del programa de seguridad: un registro de riesgos común, la asignación de controles a uno o varios marcos normativos, la recogida de evidencias para auditoría y la generación de informes para la dirección. El diagnóstico de la postura de seguridad se apoya, característicamente, en **cuestionarios y análisis de brechas** (*gap analysis*) antes que en pruebas técnicas. La plataforma española **Ulises GRC** ilustra bien este patrón: ofrece un "GAP Análisis automático" que identifica las brechas de cumplimiento "al responder cuestionarios sencillos" y mantiene actualizada la documentación asociada [27].

Para los marcos que ocupan a este trabajo, varias *suites* declaran soporte explícito. **GlobalSuite Solutions**, de origen español, se presenta como un "software GRC todo en uno" que cubre ENS, NIS2 y DORA, con módulos

de inventario de activos, gestión de riesgos, continuidad, supervisión de terceros y notificación de incidentes; su propuesta de interoperabilidad consiste en "mapear transversalmente los controles", de modo que la evidencia aportada para un marco justifique los requisitos coincidentes de otro [11]. En el plano internacional, **OneTrust** ofrece un módulo de gestión de terceros y una "asignación de evidencias entre marcos para identificar áreas superpuestas de cumplimiento normativo y eliminar duplicidades" [15]. Y **Formalize** cubre DORA y NIS2 permitiendo "la conexión entre marcos y requisitos de compliance" (ISO 27001, RGPD, NIS2), si bien su enfoque es la gestión documental: proporciona "orientación clara sobre los campos aplicables" que el usuario debe cumplimentar, más que una determinación automática de qué controles le corresponden [10].

En el ámbito específico del ENS, la plataforma española **Proactivanet**, de gestión de servicios TI (ITSM/CMDB), incorpora una capa de cumplimiento que, según su documentación, "facilita el cumplimiento del ENS, NIS2 y DORA" y permite explorar los controles del Esquema Nacional de Seguridad y visualizar su relación con cada regulación [19]. Por transparencia: esta correspondencia control–normativa es la fuente secundaria de la que se ha tomado el mapeo ENS↔DORA empleado en el grafo de este trabajo (véase la Sección 3.2.1).

Junto a estas *suites* de gestión conviven, por un lado, las **herramientas de seguridad operacional** (SIEM, EDR/XDR, gestión de identidades, copias de seguridad, gestión del riesgo de terceros), que ejecutan controles técnicos en tiempo real pero no evalúan el cumplimiento normativo, y, por otro, los **recursos públicos de autodiagnóstico**, como las herramientas de autoevaluación de INCIBE o las guías CCN-STIC de implantación y verificación del ENS, que ofrecen una vía gratuita, de nuevo basada en cuestionario y lista de comprobación, para el análisis de brechas inicial.

El rasgo común a todo este segmento es que la "interoperabilidad" que ofrecen es un **mapeo estático de controles orientado a reutilizar evidencias** y a evitar trabajo duplicado entre marcos. Es una capacidad de gestión valiosa, pero distinta de la que persigue este trabajo, como se argumenta en la Sección 3.1.3.

3.1.2. Enfoques basados en grafos de conocimiento (*GraphRAG*)

Frente al enfoque de las *suites* comerciales, la investigación reciente ha explorado una vía distinta para el cumplimiento normativo asistido por inteligencia artificial: representar la regulación como un **grafo de conocimiento** que estructure sus referencias cruzadas y sirva de anclaje al razonamiento de un modelo de lenguaje, en lugar de entregarle texto plano. Esta línea,

emparentada con el paradigma *GraphRAG* (*Graph Retrieval-Augmented Generation*) [7], responde a una limitación conocida de los sistemas RAG convencionales [13]: cuando un modelo recupera fragmentos de texto regulatorio sin estructura, tiende a perder las dependencias entre artículos y a producir respuestas poco trazables.

Dos trabajos ilustran el estado de esta investigación. **GraphCompliance** [5] descompone el texto normativo en un "Policy Graph" y el contexto operativo de la organización en un "Context Graph", y alinea ambos para fundamentar el dictamen de un LLM "juez"; sobre 300 escenarios reales del RGPD, los autores reportan una mejora de "4.1-7.2 percentage points" en micro-F1 frente a las líneas base de solo-LLM y RAG, con menos falsos positivos.

Más directamente relacionado con las decisiones técnicas de este trabajo, Ershov [9] construye un **grafo de conocimiento regulatorio ejecutable** (*compliance as code*) para la normativa del Abu Dhabi Global Market, explicitando que el grafo se "almacena en una base de datos Neo4j" y se interroga mediante *Cypher*.

De ambos trabajos se desprende una conclusión común: estructurar la norma como grafo preserva las relaciones internas del corpus regulatorio y proporciona al modelo citas textuales concretas, lo que mejora la trazabilidad de sus respuestas.

El módulo del Sistema Técnico se inscribe en esta corriente, pero da un paso más en el reparto de responsabilidades entre el grafo y el modelo de lenguaje. Mientras los trabajos citados emplean el grafo principalmente como *contexto* que un LLM razona, el módulo del ST traslada al propio grafo la **lógica de inferencia normativa** (los suelos del PCE-NIS2, los refuerzos de DORA y el Principio de Máximos), de modo que la determinación de los requisitos aplicables se resuelve de forma **determinista** mediante una consulta, y el modelo de lenguaje queda reservado a la interacción en lenguaje natural con el usuario. Esta separación, que el capítulo de Diseño desarrolla, es la que motiva la elección de la tecnología de grafo.

Justificación de Neo4j

Para soportar este modelo se ha optado por Neo4j, la base de datos de grafos de propósito general más consolidada y la empleada por la literatura de grafos regulatorios antes citada [9]. La elección se sostiene en tres razones. Primera, su **modelo de grafo de propiedades** [22] permite almacenar atributos tanto en los nodos como en las relaciones; esto es decisivo para este trabajo, porque buena parte de la lógica normativa (los suelos y exclusiones que el PCE-NIS2 impone a cada control) no pertenece al control en sí, sino a su interpretación bajo una normativa, y se modela por ello como

propiedades de la relación correspondiente (almacenar las reglas normativas en las relaciones del grafo, y no en los nodos, es lo que en el capítulo de Diseño se denomina "grafo gordo"). Segunda, su lenguaje de consulta declarativo, ***Cypher***, permite expresar el recorrido del grafo y el cálculo del Principio de Máximos como una única consulta de inferencia. Y tercera, su soporte nativo de **índices vectoriales** habilita la búsqueda semántica sobre los nodos mediante *embeddings* [21], integrando en una misma plataforma el grafo estructurado y la recuperación por similitud que el agente necesita.

***Neo4j Aura* y la instancia gratuita.**

En el plano práctico, Neo4j se ofrece como servicio gestionado en la nube a través de ***Neo4j Aura***, lo que evita administrar infraestructura propia. Su nivel gratuito, **AuraDB Free**, ha sido suficiente para el desarrollo de este trabajo: proporciona "una base de datos de grafos plenamente funcional" sin límite temporal, "gratis para siempre", con una capacidad máxima de "50.000 nodos y 175.000 relaciones" [14]. Dado que el grafo de conocimiento del módulo del Sistema Técnico (los controles y requisitos del ENS junto con los artículos de NIS2 y DORA) es del orden de unos pocos miles de nodos, encaja holgadamente dentro de esos límites. La principal restricción de este nivel es que la instancia se pausa automáticamente "tras tres días de inactividad" [14], una limitación asumible en el contexto de un prototipo académico. Aura aloja únicamente el grafo de conocimiento, común a todas las evaluaciones y carente de datos de empresa: ni el perfil de la organización ni su interacción con la herramienta se almacenan en la nube, y la inferencia corre sobre modelos *open-weight* autoalojados, sin recurrir a ninguna API comercial de terceros.

3.1.3. Posicionamiento del módulo del Sistema Técnico

De la revisión anterior se desprende que las plataformas GRC resuelven con solvencia la **gestión** del cumplimiento (documentación, evidencias, informes) y que su tratamiento de la concurrencia de marcos se limita a un mapeo estático de controles para reaprovechar evidencias. Ninguna de las herramientas analizadas modela los elementos que hacen no trivial la interoperabilidad descrita en la Sección 3.3: los suelos mínimos diferenciados que el PCE-NIS2 impone a entidades esenciales e importantes, su expresión global o dimensional sobre CITAD, o los refuerzos innegociables de DORA. En consecuencia, la determinación de **qué requisitos aplican exactamente a una organización concreta, y a qué nivel**, sigue siendo una tarea experta y manual: la herramienta gestiona el resultado, pero no lo infiere.

Es precisamente ahí donde se posiciona el módulo del Sistema Técnico.

No pretende ser otra *suite* GRC (no administra el ciclo de vida de las evidencias ni genera expedientes de auditoría), sino una capa previa de **inferencia normativa**: a partir del perfil de la empresa, resuelve automáticamente la confluencia de ENS, NIS2 y DORA mediante el Principio de Máximos y entrega el conjunto consolidado de requisitos exigibles. La viabilidad técnica de esa inferencia descansa en la representación del problema como un grafo de conocimiento, cuyos fundamentos se han revisado en la Sección 3.1.2.

3.2. Marco normativo europeo y nacional

El módulo del Sistema Técnico opera sobre la confluencia de tres marcos de ciberseguridad de obligado cumplimiento: dos de origen europeo, DORA y la Directiva NIS2, y uno nacional, el Esquema Nacional de Seguridad. Aunque persiguen un objetivo común, difieren en su ámbito de aplicación, en la forma de graduar sus exigencias y en su grado de madurez normativa. Esta sección caracteriza cada uno; la Sección 3.3 aborda su interoperabilidad.

Antes de entrar en cada marco conviene explicitar una premisa que vertebra todo el diseño del sistema y que justifica por qué el ENS ocupa una posición central. De los tres, el ENS es el único que ofrece un **catálogo de controles técnicos concretos y verificables** (las medidas de seguridad de su Anexo II) acompañado, además, de una **metodología oficial de comprobación del cumplimiento**: la guía CCN-STIC-808, *Verificación del cumplimiento* [4]. La NIS2 y DORA, en cambio, formulan sus exigencias en un plano más alto (obligaciones de gestión, de gobernanza y de resultado) sin descender a una lista cerrada de controles técnicos directamente auditables. El ENS responde así a *cómo* se comprueba el cumplimiento; NIS2 y DORA establecen *qué* debe lograrse.

De esa asimetría nace la decisión metodológica que sostiene el motor de inferencia: **adoptar el ENS como sustrato técnico común**, el denominador sobre el que se expresan, en forma de controles y refuerzos evaluables, las obligaciones de NIS2 y de DORA. Esta proyección no es una construcción arbitraria de este trabajo: el propio Centro Criptológico Nacional la ha seguido oficialmente para la NIS2, al publicar el PCE-NIS2 (CCN-STIC-892) como correspondencia entre la Directiva y las medidas del ENS [3]. Para DORA no existe todavía un mapeo oficial equivalente, pues su aplicación es muy reciente (desde enero de 2025), lo que obliga, como se detalla en la Sección 3.2.1, a recurrir a una correspondencia de fuente secundaria. Es precisamente esta capa técnica común y verificable la que hace posible combinar los tres marcos mediante el Principio de Máximos, que se desarrolla en la Sección 3.3.

3.2.1. DORA: sector financiero y proveedores TIC críticos

El Reglamento (UE) 2022/2554, conocido como DORA (*Digital Operational Resilience Act*), establece un marco común para la **resiliencia operativa digital del sector financiero** de la Unión [16]. Frente a la Directiva NIS2, que requiere transposición a los ordenamientos nacionales, DORA es un **Reglamento**: de aplicación directa y uniforme en todos los Estados miembros, sin necesidad de norma interna de desarrollo, y aplicable desde el **17 de enero de 2025** [16] (art. 64).

Su **ámbito de aplicación** (art. 2) abarca a un amplio conjunto de **entidades financieras** (entre otras, entidades de crédito, empresas de servicios de inversión, gestoras de fondos, entidades de seguros y reaseguros, entidades de pago y de dinero electrónico, depositarios centrales de valores o entidades de contrapartida central) y, de forma novedosa, a los **proveedores terceros de servicios TIC** que dan soporte a dichas entidades.

El Reglamento se articula en torno a cinco bloques temáticos o pilares:

- **Gestión del riesgo relacionado con las TIC**, que obliga a las entidades a dotarse de un marco de gobernanza y control interno que garantice una gestión eficaz del riesgo TIC.
- **Gestión, clasificación y notificación de incidentes relacionados con las TIC**, que armoniza el tratamiento de los incidentes y su comunicación a las autoridades competentes.
- **Pruebas de resiliencia operativa digital**, que exige evaluaciones periódicas, incluidas, para las entidades de mayor criticidad, pruebas de penetración basadas en amenazas.
- **Gestión del riesgo de terceros TIC**, que regula la relación contractual con los proveedores y, de manera destacada, instaura un **marco de supervisión de los proveedores terceros esenciales** de servicios TIC, sometidos a vigilancia directa por las Autoridades Europeas de Supervisión [16] (considerandos 28-31).
- **Intercambio de información**, que habilita acuerdos voluntarios de puesta en común de inteligencia sobre amenazas entre entidades financieras.

DORA incorpora también un **principio de proporcionalidad** (art. 4): la aplicación de sus requisitos debe ponderarse en función del tamaño y el perfil de riesgo general de la entidad, así como de la naturaleza, la escala y la complejidad de sus servicios, actividades y operaciones. No obstante, y este es el matiz decisivo a efectos del motor de inferencia, esta proporcionalidad

no se traduce en una graduación por niveles o categorías equiparable a la del ENS o a los suelos del PCE-NIS2: cuando una exigencia de DORA resulta aplicable a una entidad, opera como un **refuerzo innegociable**, no como un mínimo modulable por la categoría del sistema. Esta diferencia de mecánica (exigencia transversal por aplicabilidad sectorial frente a suelos graduados por categoría y dimensión) condiciona el modo en que las tres normas deben combinarse.

Por último, el propio Reglamento se declara ***lex specialis*** respecto a la Directiva NIS2 [16] (considerando 16): en lo relativo a la gestión del riesgo TIC y a la notificación de incidentes, las entidades financieras sujetas a DORA quedan reguladas por este y no por la NIS2. Esta regla de precedencia opera al delimitar qué marcos resultan aplicables a la entidad, con carácter previo a la evaluación técnica.

Alcance de DORA en este trabajo

Es necesario delimitar desde ahora hasta qué punto el módulo del Sistema Técnico operacionaliza el Reglamento, pues su tratamiento es deliberadamente parcial. A diferencia de la NIS2, que cuenta con un perfil de cumplimiento oficial (el PCE-NIS2 del CCN) que proyecta sus exigencias sobre las medidas del ENS, **DORA carece de un perfil equivalente publicado por la autoridad nacional**. Por ello, el sistema no evalúa el Reglamento en su integridad, sino la parte de él que admite una correspondencia razonable con los controles del ENS: fundamentalmente el pilar de **gestión del riesgo relacionado con las TIC** (gobernanza, marco y políticas de seguridad de la información; arts. 5, 6, 9 y 10), junto con previsiones puntuales sobre gestión de incidentes (art. 17) y riesgo de terceros (art. 28). Estas exigencias se han modelado como **refuerzos sobre controles ENS** que se activan cuando la entidad queda sujeta a DORA, a partir de una correspondencia ENS↔DORA de **fuente secundaria no oficial** [19], al no existir un *crosswalk* normativo de referencia.

Quedan, en consecuencia, **fuera del ámbito evaluable** del sistema los aspectos de DORA que no tienen reflejo directo en el catálogo del ENS: las **pruebas avanzadas de resiliencia**, el **marco de supervisión de los proveedores terceros esenciales**, de naturaleza institucional y ajeno a la evaluación de una entidad concreta, y los **plazos formales de notificación de incidentes**. Esta acotación, y la decisión de modelar DORA como capa de refuerzo sobre el ENS en lugar de como marco autónomo, se justifican en el capítulo de Diseño y se retoman como línea de trabajo futuro en el capítulo de Conclusiones.

3.2.2. Directiva NIS2: sectores esenciales e importantes

La Directiva (UE) 2022/2555 (NIS2) persigue elevar el nivel común de ciberseguridad en la Unión, ampliando el conjunto de sectores y entidades sujetos a obligaciones respecto a la directiva anterior [17]. Las entidades incluidas en su ámbito se clasifican en dos categorías, **esenciales** e **importantes**, en función de la criticidad de su sector y de su tamaño, sometidas a regímenes de supervisión distintos: *ex ante* para las esenciales y *ex post* para las importantes [3].

De las obligaciones que la Directiva impone, tres resultan especialmente relevantes para la evaluación técnica que aborda este trabajo:

- **Artículo 20: Gobernanza.** Los órganos de dirección de las entidades esenciales e importantes deben aprobar las medidas de gestión de riesgos de ciberseguridad, supervisar su puesta en práctica y responder por el incumplimiento.
- **Artículo 21: Medidas para la gestión de riesgos de ciberseguridad.** Obliga a adoptar medidas técnicas, operativas y de organización adecuadas y proporcionadas. Su apartado 2 enumera un conjunto de medidas concretas, de la letra a) a la j), que abarcan, entre otras, las políticas de análisis de riesgos, la gestión de incidentes, la continuidad de la actividad, la seguridad de la cadena de suministro, el control de accesos o la autenticación multifactor.
- **Artículo 23: Obligaciones de notificación.** Exige notificar a las autoridades competentes o al CSIRT (*Computer Security Incident Response Team*, Equipo de Respuesta a Incidentes de Seguridad Informática) los incidentes con impacto significativo, con plazos estrictos (alerta temprana en 24 horas, notificación en 72 horas e informe final en un mes).

En España, el cumplimiento de la NIS2 se articula a través del ENS. El Centro Criptológico Nacional ha publicado a tal efecto la guía **CCN-STIC-892, Perfil de Cumplimiento Específico (PCE-NIS2)** [3][1], que analiza los requisitos de la Directiva, en particular los artículos 20 y 21, en relación con las medidas de seguridad del ENS (Anexo II), identificando qué

[1]Durante la elaboración de esta memoria, la versión de la guía CCN-STIC-892 aquí empleada fue retirada del portal del CCN-CERT. El CCN se encuentra elaborando una nueva versión del Perfil de Cumplimiento Específico, orientada a las entidades en el ámbito del *Reglamento de Ejecución (UE) 2024/2690* y a la espera de la transposición de la Directiva NIS2 al ordenamiento español [2]; las certificaciones del ENS basadas en el perfil retirado conservan su validez. El motor de inferencia descrito en este trabajo se sustenta en la versión de 2024, vigente en el momento de su desarrollo.

medidas resultan necesarias y en cuáles se exigen refuerzos. Este trabajo toma como base esta correspondencia para los artículos **20 a 21.2 j)** de la Directiva. Las obligaciones de notificación del **artículo 23**, de naturaleza procedimental y sin reflejo en el catálogo de controles del Anexo II del ENS, quedan, por ello, fuera del alcance evaluable del sistema, al igual que sucede con los plazos de notificación de DORA (Sección 3.2.1).

La aportación decisiva del PCE-NIS2 a efectos del motor de inferencia es que establece, sobre las medidas del ENS, **suelos mínimos de exigencia** diferenciados para entidades esenciales e importantes, así como **exclusiones** (medidas que la NIS2 no impone) y **refuerzos** adicionales. Estos suelos pueden expresarse de forma global (sobre la categoría del sistema) o dimensional (sobre una dimensión de seguridad concreta), lo que enlaza directamente con la complejidad descrita en la Sección 3.2.3 y con el Principio de Máximos de la Sección 3.3.

3.2.3. Esquema Nacional de Seguridad (ENS)

El Esquema Nacional de Seguridad, regulado en España por el Real Decreto 311/2022, establece la política de seguridad para la utilización de medios electrónicos por las entidades de su ámbito de aplicación, que comprende a todo el sector público y a las organizaciones del sector privado que le prestan servicios o le suministran tecnología [20].

El elemento central del ENS a efectos de este trabajo es su sistema de **categorización**, definido en el Anexo I del Real Decreto. La categoría de un sistema de información se determina valorando el impacto que tendría sobre la organización un incidente que afectara a la seguridad de la información tratada o de los servicios prestados, atendiendo a cinco **dimensiones de seguridad**, identificadas por sus iniciales en mayúscula [20] (Anexo I, ap. 2):

- **Confidencialidad [C]**
- **Integridad [I]**
- **Trazabilidad [T]**
- **Autenticidad [A]**
- **Disponibilidad [D]**

Estas cinco dimensiones, que a lo largo de este trabajo se abrevian conjuntamente como **CITAD**, constituyen la base sobre la que se evalúa todo el cumplimiento técnico.

Cada dimensión se valora en uno de tres **niveles de seguridad** (BAJO, MEDIO o ALTO), según la gravedad del perjuicio que un incidente causaría

sobre las funciones de la organización, sus activos o los individuos afectados (perjuicio limitado, grave o muy grave, respectivamente) [20] (Anexo I, ap. 3). Cuando un sistema trata distintas informaciones o presta distintos servicios, el nivel de cada dimensión es el mayor de los establecidos para cada uno de ellos.

A partir de los niveles dimensionales se determina la **categoría de seguridad** global del sistema, que puede ser **BÁSICA, MEDIA o ALTA** [20] (Anexo I, ap. 4):

- **ALTA**, si alguna de sus dimensiones alcanza el nivel ALTO;
- **MEDIA**, si alguna alcanza el nivel MEDIO y ninguna uno superior;
- **BÁSICA**, si alguna alcanza el nivel BAJO y ninguna uno superior.

Existe un matiz de esta norma que resulta decisivo para el diseño del sistema y que se retoma en la Sección 3.3: la determinación de la categoría global **no altera el nivel de seguridad de las dimensiones que no han influido en ella** [20] (Anexo I, ap. 4.2). Un sistema puede así ser de categoría global ALTA y, sin embargo, conservar una dimensión concreta, por ejemplo la Disponibilidad, en nivel BAJO. Esta coexistencia de una categoría global con niveles dimensionales independientes es el origen de buena parte de la complejidad que el motor de inferencia debe resolver.

Sobre esta categorización, el ENS define en su Anexo II un conjunto de **medidas de seguridad** cuya exigibilidad depende de la categoría del sistema y del nivel de cada dimensión, aplicándose algunas de forma reforzada en función del nivel alcanzado. La estructura concreta con la que estas medidas y sus requisitos se han modelado en el grafo de conocimiento de este trabajo se detalla en el capítulo de Diseño.

3.3. El reto de la interoperabilidad normativa

Una misma organización puede quedar sujeta simultáneamente a más de uno de los marcos descritos: una entidad financiera que además preste servicios a una administración pública estaría afectada, a la vez, por DORA, por la NIS2 y por el ENS. Dado que los tres regulan la misma materia, la seguridad de los sistemas de información, pero con exigencias de distinto origen y graduación, evaluarlos por separado conduciría a resultados redundantes o, peor, contradictorios. El sistema necesita por tanto **reglas que permitan combinar los tres marcos de forma coherente**. Se aplican dos, que operan en planos distintos: una regla de **precedencia** en el momento de clasificar a la entidad (*lex specialis*) y una regla de **agregación**

de exigencias en el momento de evaluar cada control (el **Principio de Máximos**).

La precedencia (*lex specialis*). Una primera regla determina qué marcos resultan aplicables a la entidad: cuando una organización financiera queda sujeta a DORA, este prevalece sobre la NIS2 en lo que ambos regulan [16] (considerando 16). Esta determinación de aplicabilidad se resuelve en una fase previa de clasificación de la entidad, anterior a la evaluación técnica, de modo que el agente técnico recibe ya delimitado el alcance a evaluar. El reto que aborda el motor técnico es la segunda regla, la de agregación de exigencias.

La agregación: el Principio de Máximos

Resuelta la aplicabilidad, un mismo control técnico del ENS puede recibir exigencias de varias procedencias a la vez: el nivel que la propia organización requiere según su categorización ENS, el **suelo mínimo** que el PCE-NIS2 impone a las entidades esenciales o importantes (Sección 3.2.2) y los **refuerzos** que DORA exige al sector financiero (Sección 3.2.1). El Principio de Máximos resuelve este concurso con un criterio único: **gana siempre la exigencia más restrictiva**. Formalmente, el nivel de evaluación de cada control es el **máximo** entre el nivel derivado del riesgo propio de la entidad y los suelos legales que le resulten aplicables. La justificación es doble y conservadora: la ley puede **elevar** la seguridad exigida por encima de lo que la organización habría adoptado por sí sola, pero **nunca rebajarla** por debajo de su propio nivel de riesgo.

Este criterio se complica por el matiz **global frente a dimensional** introducido en la Sección 3.2.3. Los suelos legales del PCE-NIS2 pueden expresarse de dos formas: de manera **global**, referida a la categoría del sistema en su conjunto (BÁSICA/MEDIA/ALTA), o de manera **dimensional**, referida al nivel de una dimensión concreta de las CITAD. Además, el propio ENS admite que un sistema de categoría global ALTA conserve, por ejemplo, la Disponibilidad en un nivel inferior, lo que en rigor permitiría modular la exigencia dimensión a dimensión. Esta coexistencia de una magnitud global con niveles dimensionales independientes es una de las principales fuentes de complejidad del problema; el modo concreto en que el motor de inferencia la aborda se detalla en el capítulo de Diseño (Sección 4.8.3).

La aportación de DORA al Principio de Máximos es de naturaleza distinta a la de la NIS2. Como se expuso en la Sección 3.2.1, DORA no introduce suelos graduados por categoría, sino **refuerzos innegociables**: cuando la entidad está sujeta a DORA, los controles ENS que el mapeo asocia al Reglamento se incorporan a la evaluación con sus refuerzos exigidos **con independencia de la categoría** del sistema. En términos del principio,

DORA no compite por ser el máximo de un nivel graduado, sino que **añade obligaciones** que se auditan en todo caso.

Finalmente, una vez determinado el nivel ganador de cada control, el sistema completa la composición **inyectando los refuerzos requeridos** (los controles adicionales que la NIS2 o DORA exijan sobre el de partida) y **descartando las exclusiones** que el PCE-NIS2 contempla (medidas del ENS que la NIS2 expresamente no impone a una entidad esencial o importante). El resultado es, para cada control, un conjunto consolidado de requisitos que integra de forma única las tres normativas. La manera concreta en que esta lógica se ha modelado sobre el grafo de conocimiento, y se ha trasladado a una consulta de inferencia, constituye el núcleo del capítulo de Diseño.

Capítulo 4

Análisis funcional y diseño del módulo del Sistema Técnico (ST)

Este capítulo expone la arquitectura y el funcionamiento interno del módulo del Sistema Técnico. En primer lugar, se analiza el flujo de evaluación completo, desde la recepción del perfil de la empresa hasta la emisión del veredicto, prestando especial atención a las dos etapas fundamentales del motor: el descubrimiento semántico de controles y la inferencia de requisitos mediante el Principio de Máximos. Una vez establecido este modelo funcional, se detalla el diseño técnico adoptado, abordando la estructura distribuida de los servicios, la construcción del grafo de conocimiento y el modelado de la consulta que sustenta el razonamiento normativo.

4.1. Descripción del flujo de evaluación y contexto de uso

El módulo del Sistema Técnico no se invoca de forma aislada. Dentro de la arquitectura de *Ciber-AsesorIA*, el Coordinador, tras obtener del Sistema Normativo la lista de **controles legislativos** aplicables al perfil de la empresa, abre una conversación con el módulo del ST **por cada control**. El módulo del ST recibe, por tanto, una unidad de trabajo bien delimitada: un control legislativo (identificador, título y descripción) y el perfil de la empresa. Su cometido es transformar esa entrada en un **veredicto técnico razonado y trazable**.

El recorrido, de control legislativo recibido a veredicto emitido, que el diagrama de actividad del Anexo E ilustra al completo, se organiza en tres

etapas:

1. **Preparación (primer turno).** El sistema resuelve, mediante las dos fases descritas en las Secciones 4.2 y 4.3, **qué requisitos técnicos concretos** debe auditar para ese control legislativo y ese perfil. El resultado es una lista cerrada de requisitos, ya filtrada y consolidada entre ENS, NIS2 y DORA.
2. **Auditoría conversacional (turnos siguientes).** El sistema recorre los requisitos técnicos **uno a uno**: por cada requisito formula una pregunta en lenguaje de negocio y recoge la respuesta del usuario. No se evalúa nada hasta haber recabado la información de todos los requisitos de un control.
3. **Emisión del veredicto.** Completados los requisitos de un control técnico, el sistema decide si **cumple** o **no cumple**; agregados todos los controles técnicos de ese control legislativo, el módulo del ST emite **su veredicto para dicho control legislativo**, una puntuación numérica (1–10) y una traza de auditoría.

Una particularidad del flujo es la ***caché* de veredictos de sesión**: como un mismo control técnico puede resultar pertinente para varios controles legislativos distintos, una vez evaluado no se vuelve a preguntar al usuario por él, sino que se reutiliza el veredicto ya emitido (véase la Sección 4.4). Esto evita la repetición y mantiene la coherencia de la evaluación a lo largo de toda la sesión.

4.2. Diseño funcional de la evaluación (Fase 1: descubrimiento semántico)

Un control legislativo llega al módulo del ST como **texto libre** (título y descripción), sin indicar qué control técnico del ENS lo materializa. La primera fase resuelve esa correspondencia: **descubrir, entre todo el catálogo de controles técnicos, aquellos que permiten auditar el artículo recibido.**

Vectorización

El título y la descripción del control legislativo se combinan en un único texto y se transforman en un vector mediante un modelo de *embeddings* (`mxbai-embed-large`), el mismo con el que previamente se vectorizaron los controles técnicos del grafo. Trabajar en el mismo espacio vectorial permite comparar la intención del artículo con el significado de cada control.

Umbral de similitud

Sobre el grafo se calcula la **similitud del coseno** entre el vector de consulta y el vector de cada control técnico, y se retienen como candidatos únicamente aquellos cuya similitud supera un **umbral de 0,65**. El umbral actúa como primer filtro de ruido: descarta los controles claramente ajenos al artículo, pero es deliberadamente permisivo para no perder candidatos pertinentes (los falsos positivos restantes los depura el paso siguiente).

Búsqueda restringida a controles base del ENS, y exclusión de refuerzos

La búsqueda semántica opera **únicamente sobre los controles base del ENS**, no sobre nodos de NIS2 ni de DORA. Esto es coherente con la decisión metodológica adoptada en el Capítulo 3: el ENS se emplea como **sustrato técnico común** y las exigencias de NIS2 y DORA no se buscan de forma autónoma, sino que se **anclan a los controles del ENS mediante los mapeos de correspondencia** entre marcos. De ahí que en el grafo las obligaciones de NIS2 y DORA se materialicen como refuerzos y relaciones colgados de un control ENS base, y no como objetivos de búsqueda independientes.

En consecuencia, la búsqueda excluye explícitamente los controles que son **refuerzo** de otro (los que mantienen una relación de refuerzo hacia un control base). La razón es funcional: un refuerzo no es un objetivo de evaluación autónomo, sino una exigencia adicional que se activa sobre su control base cuando el nivel de seguridad, o un suelo impuesto por NIS2 o DORA, así lo requiere. Por ello, la Fase 1 solo descubre **controles base**; sus refuerzos, incluidos los que activan NIS2 y DORA a través de los mapeos, se incorporan, si procede, en la Fase 2.

Filtrado de falsos positivos (juez)

La similitud vectorial es necesaria pero no suficiente: dos controles pueden compartir vocabulario de seguridad y, sin embargo, evaluar prácticas distintas. Para depurar estos **falsos positivos**, los candidatos se someten a un **juez basado en un modelo de lenguaje**, al que se le proporciona el control legislativo, el contexto de la empresa y la lista de candidatos, y que selecciona únicamente aquellos cuya auditoría aportaría **evidencia directa** de cumplimiento del artículo. El juez no devuelve un número fijo de controles, sino los que considera pertinentes; el sistema aplica un tope de seguridad y, ante cualquier fallo del juez, recurre a los candidatos de mayor similitud. La salida de la Fase 1 es, así, una lista depurada de controles base realmente pertinentes.

4.3. Motor de inferencia de requisitos (Fase 2: Principio de Máximos)

Para cada control base seleccionado en la Fase 1, la segunda fase determina **qué requisitos técnicos concretos deben auditarse y con qué nivel de exigencia**, resolviendo de forma automática la confluencia de ENS, NIS2 y DORA descrita en el Capítulo 3.

Cálculo del nivel ganador

El nivel de exigencia con el que se evalúa un control no es único: puede provenir del riesgo propio de la empresa (su nivel de seguridad ENS en las dimensiones que el control afecta) y de los **suelos mínimos** que el PCE-NIS2 impone según el tipo de entidad (esencial o importante), expresados de forma global o dimensional. El motor aplica el **Principio de Máximos**: el nivel ganador es el **máximo** entre el riesgo propio y los suelos legales aplicables. A efectos de cálculo, los niveles se ordenan numéricamente (NINGUNO = 0; BÁSICA/BAJO = 1; MEDIA/MEDIO = 2; ALTA/ALTO = 3), de modo que el máximo se reduce a una comparación.

Requisitos base

Una vez fijado el nivel ganador, se seleccionan los requisitos del control que resultan exigibles a ese nivel. Cada requisito lleva asociado el nivel a partir del cual el ENS lo exige, su *categoría de aplicación*, de modo que solo se incluyen aquellos cuyo umbral queda alcanzado por el nivel ganador; los requisitos propios de niveles superiores no llegan a activarse. Cada requisito se acompaña de sus **orígenes**: la lista de normativas (ENS, NIS2, DORA) que justifican su exigencia, lo que garantiza la trazabilidad del resultado.

Refuerzos y exclusiones

Sobre el conjunto base, el motor **incorpora los refuerzos** que correspondan, ya porque el nivel ganador los active, ya porque la NIS2 o DORA los exijan explícitamente, y **descarta las exclusiones** que el PCE-NIS2 contempla para entidades esenciales o importantes. La aportación de DORA es de naturaleza distinta: no compite por elevar un nivel graduado, sino que **añade refuerzos innegociables** que se auditan con independencia de la categoría del sistema cuando la entidad está sujeta al Reglamento.

El resultado de la Fase 2 es, para cada control, un **conjunto consolidado de requisitos** que integra de forma única las tres normativas. La manera en que esta lógica se ha modelado sobre el grafo y se ha trasladado

a una única consulta de inferencia se detalla más adelante en este mismo capítulo.

4.4. Modelo de datos

El sistema maneja un pequeño número de estructuras de datos bien definidas que circulan entre los subsistemas y dentro del propio motor.

`ControlItem`: la unidad de trabajo.

Es la estructura con la que el Sistema Normativo entrega cada control legislativo al módulo del ST a través del Coordinador. Contiene el identificador del control, la normativa de procedencia, el título y la descripción. Es la entrada de todo el flujo descrito en la Sección 4.1.

`CompanyProfile`: el perfil que parametriza la inferencia.

Recoge la aplicabilidad regulatoria de la empresa (sujeción a DORA, a NIS2 con su tipo esencial/importante/ninguno, y al ENS), datos de contexto (sector y tamaño) y, procedente del componente de clasificación CITAD, la **categoría global** y el **nivel de cada dimensión** de seguridad. Este perfil es el que, en la Fase 2, determina los suelos legales y el riesgo propio que alimentan el Principio de Máximos.

Requisito técnico

Cada requisito a auditar se describe mediante: su **descripción** en lenguaje de negocio, su **tipo**, la marca de **esencial** y sus **orígenes** normativos. El tipo distingue dos formas de cumplimiento, decisivas para el criterio de auditoría:

- **Evidencia:** la empresa debe disponer de **algún registro o documento** que lo respalde. No se exige formalidad (una hoja de cálculo, un correo archivado o un ticket bastan); solo se incumple ante la ausencia total de rastro documental.
- **Verificación:** la empresa debe **demostrar que la práctica se realiza** realmente; no se exige documentación, basta con describir cómo y con qué alcance se lleva a cabo.

La marca **esencial**, tomada de la metodología de verificación del ENS de la guía CCN-STIC-808, que señala en cada medida los requisitos conside-

rados esenciales [4], introduce una regla de agregación, aplicada en el juicio de cumplimiento de cada control y reflejada de forma determinista en la puntuación final: si un requisito esencial no se cumple, el control completo se considera **no cumplido**, con independencia del resto de requisitos.

Veredictos

Cada control técnico evaluado produce un veredicto interno (`cumple` / `no cumple`, con su razón). Al cerrarse la evaluación, el sistema devuelve al usuario una respuesta que incluye el texto conversacional, una marca de finalización, el veredicto de cumplimiento, una **puntuación de 1 a 10** y las observaciones.

Caché de veredictos.

Para evitar preguntar dos veces por el mismo control técnico, el agente mantiene una ***caché* de sesión** indexada por el identificador del control base, que almacena su veredicto, la razón y el control legislativo que lo originó. La *caché* **persiste durante toda la vida de la sesión**, sin reiniciarse entre controles legislativos, para poder reutilizar veredictos; ante una coincidencia (*cache hit*), el sistema reaprovecha el resultado sin volver a interrogar al usuario.

4.5. Satisfacción de los requisitos no funcionales

Los cinco requisitos no funcionales definidos en la Sección 2.4 (RNF-01 a RNF-05) se traducen aquí en mecanismos concretos de diseño. Tres de ellos se resuelven con piezas específicas que se describen a continuación: el determinismo del cálculo (RNF-01), la trazabilidad y auditoría del veredicto (RNF-02) y la resiliencia frente a la latencia de los modelos (RNF-03). Los dos restantes descansan en decisiones de diseño tratadas en otras secciones: la extensibilidad (RNF-04) se apoya en el diseño del grafo de conocimiento (Sección 4.7), que permite añadir o modificar normativas, controles y mapeos sin reescribir el núcleo de inferencia, y la soberanía del dato (RNF-05) en la ejecución local de los modelos de pesos abiertos sobre infraestructura propia, sin dependencia de APIs de terceros.

Determinismo (RNF-01)

El motor de inferencia (el cálculo del Principio de Máximos en la Fase 2) y la **puntuación final** (1–10) son **plenamente deterministas**: son

funciones puras del perfil de la empresa y del resultado de cada requisito, sin intervención de azar. Igualmente determinista es la clasificación CITAD, resuelta por un formulario de selección y la regla del máximo. Las partes apoyadas en modelos de lenguaje (la selección del juez y el juicio de cumplimiento de cada requisito) se ejecutan con **temperatura cero** para maximizar la reproducibilidad, aunque no constituyen, en sentido estricto, un proceso determinista.

Trazabilidad y auditoría (RNF-02)

El resultado nunca es una caja negra. Por un lado, cada requisito evaluado conserva sus **orígenes** normativos, de modo que siempre puede justificarse por qué se exigió. Por otro, el sistema genera una **traza de auditoría** (un fichero por sesión) que registra, para cada control técnico evaluado, su veredicto y razón, la similitud que motivó su selección y los **orígenes normativos** que lo exigieron (NIS2, DORA o ENS Base/Refuerzo), además del veredicto final. Esto permite reconstruir a posteriori qué se evaluó, con qué resultado y por qué, tal como ilustra la Figura 4.1 mediante un ejemplo ilustrativo.

Latencia y *timeouts* (RNF-03).

La fase de descubrimiento utiliza, para el juez, un modelo de lenguaje de mayor capacidad y, por tanto, más lento. Para acomodar su tiempo de respuesta, el Coordinador fija un **tiempo de espera amplio (200 segundos)** en las llamadas al módulo del ST. La resiliencia se garantiza con una **degradación elegante**: si el juez no responde o falla, el sistema continúa con los candidatos de mayor similitud, evitando que un fallo del modelo bloquee la evaluación.

4.6. Arquitectura global de *Ciber-AsesorIA*

El módulo del Sistema Técnico no es una aplicación monolítica, sino uno de los servicios de la arquitectura distribuida de *Ciber-AsesorIA* presentada en la Sección 1.3. La Figura 4.2 sitúa el módulo del ST dentro del conjunto y explicita sus dependencias.

Ubicación del módulo del ST

El módulo del Sistema Técnico se expone como un servicio FastAPI (`:5001`) que el Coordinador invoca **una vez por cada control legislativo**:

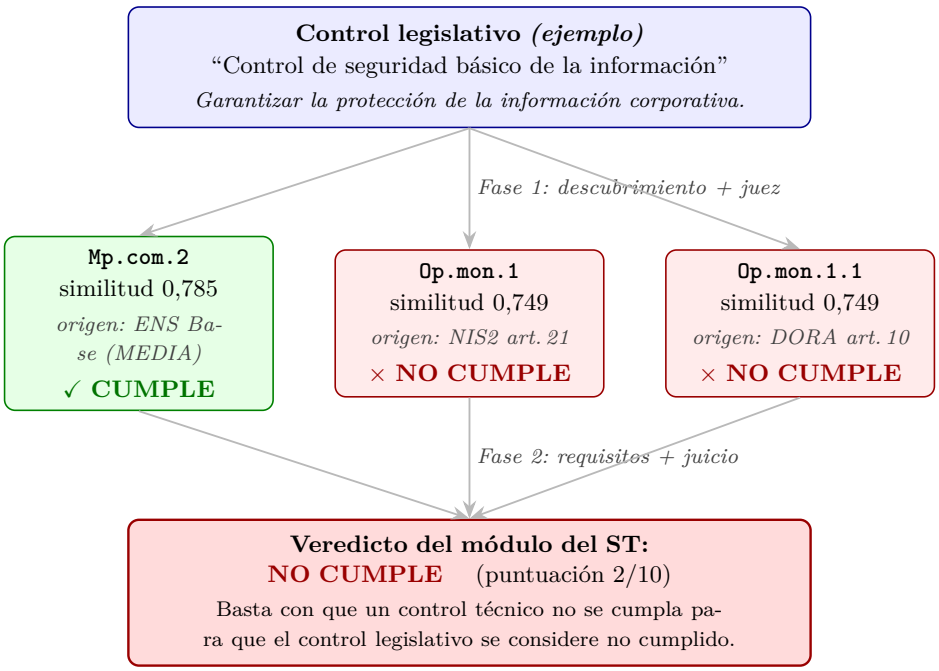

Figura 4.1: Traza de auditoría sobre un **control legislativo de ejemplo** de carácter ilustrativo, que no reproduce el texto de ningún control normativo concreto, empleado únicamente para mostrar el mecanismo de trazabilidad. La Fase 1 descubre tres controles técnicos del ENS (con su similitud semántica); cada uno indica además el **origen normativo** que lo exige (ENS, NIS2 o DORA) y recibe un veredicto individual y, por la regla de agregación, el veredicto final del control resulta *no cumple*.

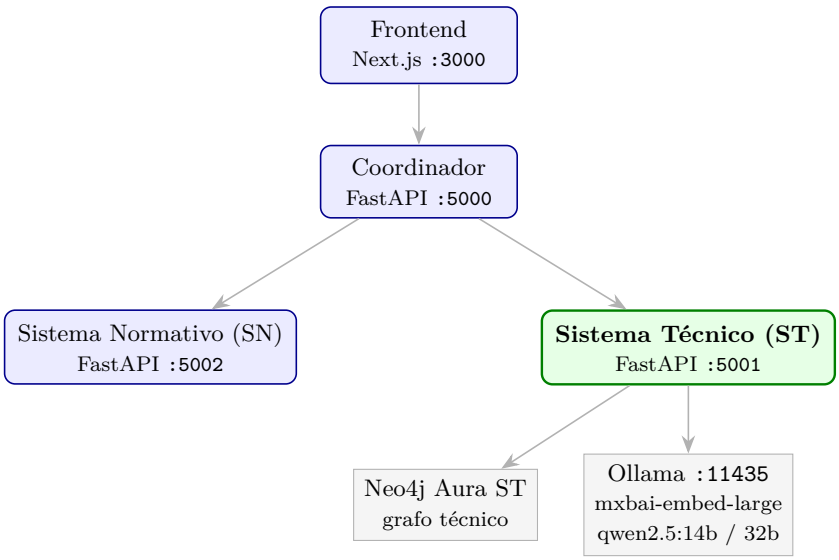

Figura 4.2: Ubicación del módulo del Sistema Técnico dentro de *CiberAsesorIA* y sus dependencias. El módulo del ST se invoca a través del Coordinador y se apoya en su propia instancia *Neo4j Aura* y en un Ollama local que sirve tanto el modelo de *embeddings* como los modelos de lenguaje.

recibe el control y el perfil de la empresa y devuelve un veredicto técnico razonado. El módulo del ST no se comunica directamente ni con el *frontend* ni con el SN; toda su interacción pasa por el Coordinador, según los contratos que se detallan en la Sección 4.12.

Dependencias externas

El módulo del ST descansa sobre dos servicios. El primero es **su propia instancia de *Neo4j Aura***, el grafo de conocimiento técnico, independiente del grafo regulatorio del SN, que se interroga mediante *Cypher* y funciones de la librería *Graph Data Science* (GDS) para la similitud vectorial. El segundo es una instancia de **Ollama**, a la que el módulo del ST se dirige como `localhost:11435`, que sirve, en una sola plataforma, el modelo de *embeddings* (`mxbai-embed-large`) y los modelos de lenguaje empleados por el agente y el juez (`qwen2.5:14b` y `32b`); su despliegue real se detalla en la Sección 5.1.

Privacidad por diseño

El razonamiento del modelo no se delega en ninguna API comercial de terceros: tanto los *embeddings* como la inferencia se realizan sobre modelos

Cuadro 4.1: Tamaño del grafo de conocimiento técnico: nodos y relaciones por tipo.

Nodo	n	Relación	n
`RequisitoENS`	1030	`EVALUA_MEDIANTE`	1054
`ControlENS`	222	`AGRUPA_CONTROL`	222
`ArticuloDORA`	68	`REFUERZA_A`	133
`ArticuloNIS2`	11	`CUMPLE_NIS2`	76
`MarcoENS`	4	`CUMPLE_DORA`	68
Total	**1335**	**Total**	**1553**

open-weight autoalojados, a los que el módulo del ST se conecta mediante un túnel cifrado (Sección 5.1). De este modo, ni el perfil de la organización ni sus respuestas se entregan a proveedores externos de IA. El único componente que reside en nube pública es el **grafo de conocimiento técnico** (*Neo4j Aura*), común a todas las evaluaciones y carente de datos de empresa, en línea con lo expuesto en la Sección 3.1.2.

4.7. Diseño del grafo de conocimiento técnico

El núcleo del módulo del ST es un grafo de conocimiento que adopta el ENS como **sustrato técnico común** (Sección 3.2) y proyecta sobre él las exigencias de NIS2 y DORA. En su estado actual, el grafo consta de **1335 nodos y 1553 relaciones**, cuyo desglose por tipo se recoge en la Tabla 4.1. Su ontología se representa en la Figura 4.3.

El diseño explota deliberadamente el modelo de **grafo de propiedades** de Neo4j que se justificó en la Sección 3.1.2: tanto los nodos como las relaciones almacenan atributos. Como se detalla en la Sección 4.7.2, buena parte de la lógica normativa no reside en los nodos sino en las relaciones (la decisión de modelado que en este trabajo se denomina **"grafo gordo"**), lo que resulta decisivo para resolver el Principio de Máximos mediante una única consulta (Sección 4.8).

4.7.1. Nodos

El grafo distingue cinco tipos de nodo, resumidos en la Tabla 4.2; la Figura 4.4 muestra un ejemplo real extraído de Neo4j. El `ControlENS` es la pieza central, el control o artículo del ENS a cumplir, nodo *padre*, y cada uno se descompone en uno o varios `RequisitoENS`, los nodos *hijo* que constituyen las preguntas concretas de la auditoría. Los `ControlENS` se agrupan, a su vez, en cuatro `MarcoENS` (cumplimiento de artículos, marco organizativo, marco operacional y medidas de protección). Por último, los nodos `ArticuloNIS2`

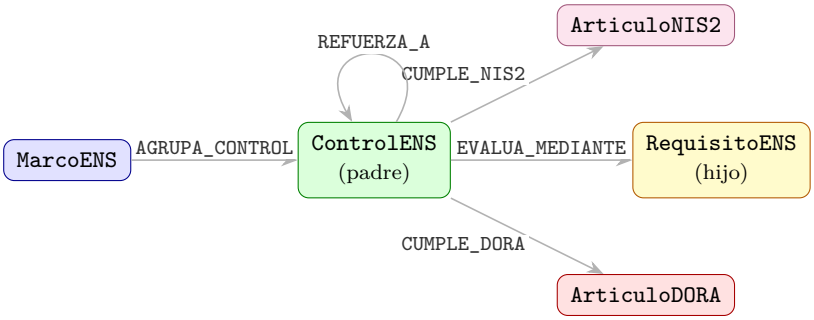

Figura 4.3: Ontología del grafo del módulo del ST: cinco tipos de nodo (`MarcoENS`, `ControlENS`, `RequisitoENS`, `ArticuloNIS2`, `ArticuloDORA`) y cinco relaciones. El `ControlENS` es el nodo central sobre el que convergen las tres normativas.

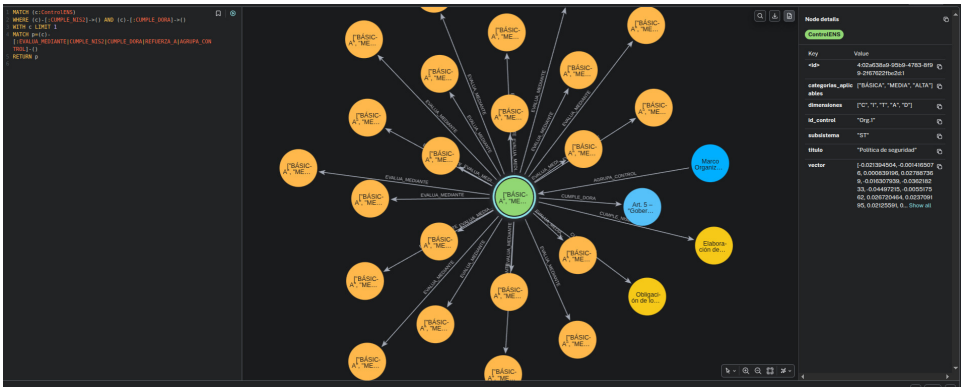

Figura 4.4: Subgrafo real extraído de Neo4j para el control `Org.1` ("Política de seguridad"): el `ControlENS` (verde, centro) enlaza con sus 22 `RequisitoENS` (`EVALUA_MEDIANTE`), su `MarcoENS` (`AGRUPA_CONTROL`), dos `ArticuloNIS2` (`CUMPLE_NIS2`) y un `ArticuloDORA` (`CUMPLE_DORA`). El panel lateral muestra sus propiedades reales (`categorias_aplicables`, `dimensiones`, `vector`).

Cuadro 4.2: Tipos de nodo del grafo de conocimiento técnico.

Nodo	Rol	Propiedades principales
`MarcoENS`	Agrupación por marco	`nombre`
`ControlENS`	Control/artículo del ENS (padre)	`id_control`, `titulo`, `categorias_aplicables`, `dimensiones`, `subsistema`, `vector`
`RequisitoENS`	Requisito a auditar (hijo)	`tipo` (Evidencia / Verificación), `descripcion`, `categorias_aplicables`, `es_esencial`, `notas`, `vector`
`ArticuloNIS2`	Artículo NIS2 (arts. 20–21.2 j)	`id_articulo`, `titulo`, `descripcion`, `embedding`
`ArticuloDORA`	Artículo/RTS de DORA	`id_dora`, `titulo`, `descripcion`, `normativa`

y **`ArticuloDORA`** representan las exigencias de cada normativa europea y se enlazan con los controles del ENS que las materializan.

Dos propiedades de estos nodos son las que gobiernan la aplicabilidad descrita en la Sección 4.3.

En el `ControlENS` y el `RequisitoENS`, `categorias_aplicables` indica a qué categorías (BÁSICA, MEDIA, ALTA) corresponde el elemento (de modo que un control padre puede ser universal pero alguno de sus requisitos hijos exigirse solo a categorías superiores), mientras que `dimensiones` registra qué dimensiones CITAD afecta cada elemento. Esta última propiedad es la que permite al motor **ponderar el riesgo propio de la empresa por dimensión**: la inferencia cruza cada control con el nivel de la organización en las dimensiones concretas que ese control afecta, tal como prescribe el Anexo II del ENS, en lugar de aplicar indiscriminadamente la categoría global (Sección 4.8.3).

Del lado de la NIS2, la decisión de expresar un suelo de forma global o dimensional la determina cuál de los suelos esté relleno en la relación (`min_global` frente a `min_nivel`), siguiendo la convención de la tabla de aplicabilidad del 892. La marca `es_esencial` del requisito implementa la regla de agregación determinista de la Sección 4.4: si un requisito esencial no se cumple, el control completo se considera no cumplido.

4.7.2. Relaciones

Las cinco relaciones del grafo se recogen en la Tabla 4.3. Tres son estructurales (`AGRUPA_CONTROL` conecta cada marco con sus controles, `EVALUA_MEDIANTE` cada control con sus requisitos y `REFUERZA_A` enlaza controles en-

Cuadro 4.3: Tipos de relación del grafo de conocimiento técnico.

Relación	**Origen → Destino**	**Propiedades**
`AGRUPA_CONTROL`	`MarcoENS` → `ControlENS`	—
`EVALUA_MEDIANTE`	`ControlENS` → `RequisitoENS`	—
`REFUERZA_A`	`ControlENS` → `ControlENS`	—
`CUMPLE_NIS2`	`ControlENS` → `ArticuloNIS2`	suelos, exclusiones y refuerzos del PCE-NIS2 ("grafo gordo")
`CUMPLE_DORA`	`ControlENS` → `ArticuloDORA`	`requiere_refuerzos`

tre sí para modelar los refuerzos, mecanismo que el propio ENS emplea) y dos son los **puentes legales** hacia las normativas europeas: `CUMPLE_NIS2` y `CUMPLE_DORA`.

El "grafo gordo"

La decisión de diseño más relevante es que las reglas que el PCE-NIS2 impone sobre cada control **no se almacenan en el control, sino en la relación `CUMPLE_NIS2`**. La razón es conceptual: la exigencia no pertenece al control en sí mismo, sino a la *interpretación* que la NIS2 hace de ese control. Así, una misma medida del ENS puede ser exigida con un nivel distinto según se mire desde el riesgo propio de la empresa o desde el suelo legal de la Directiva, y ese matiz vive en la arista (Figura 4.5).

Las propiedades de `CUMPLE_NIS2` cubren cuatro familias: **suelos mínimos**, globales (`esencial_min_global`, `importante_min_global`) o dimensionales (`esencial_min_nivel`, `importante_min_nivel`); **refuerzos** a inyectar o excluir según el tipo de entidad, **exclusiones** (`excluido_pce_nis2` y sus variantes) y **excepciones sectoriales** (`sectores_especificos`). Una última familia, `condiciones_nis2`, transporta avisos en texto para el modelo de lenguaje (notas del auditor, plazos de notificación). La relación `CUMPLE_DORA` sigue el mismo patrón pero de forma más simple: al no graduar DORA por niveles, basta con la lista de refuerzos innegociables que activa (`requiere_refuerzos`). El modo en que estas propiedades se consumen en la inferencia se detalla en la Sección 4.8.3.

4.7.3. *Embeddings* de control y similitud coseno

Sobre la estructura anterior se superpone una **capa vectorial** que habilita el descubrimiento semántico de la Fase 1 (Sección 4.8). Los `ControlENS` y los `RequisitoENS` almacenan su representación semántica en una propiedad `vector`: un vector de **1024 dimensiones** generado con el modelo `mxbai-embed-large` servido por el Ollama local.`ControlENS`.

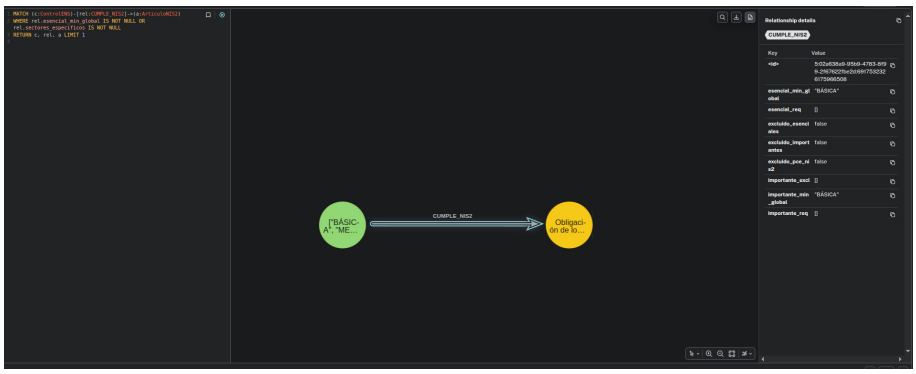

Figura 4.5: Materialización del "grafo gordo" en Neo4j: la relación `CUMPLE_NIS2` entre un `ControlENS` y un `ArticuloNIS2` almacena las reglas del PCE-NIS2 (`esencial_min_global`, `importante_min_global`, exclusiones y refuerzos) como propiedades de la propia arista, no en el nodo del control.

Vectorización previa

Los *embeddings* no se calculan en tiempo de consulta, sino **una sola vez** durante la construcción del grafo. El proceso de poblado sigue, por cada normativa, una cadena de tres etapas (ingesta de la fuente, vectorización y carga en Neo4j), de modo que el grafo ya se despliega con los vectores precomputados. Esto reserva el cómputo de *embeddings* del usuario únicamente al texto del control entrante.

Búsqueda por similitud

En cada evaluación, el agente vectoriza el control legislativo recibido y recupera los `ControlENS` más próximos calculando la **similitud coseno** entre ese vector y los del grafo mediante la función `gds.similarity.cosine` de la librería Graph Data Science. El resultado es una lista de controles candidatos ordenados por proximidad semántica, que constituye la entrada de la fase de filtrado descrita en la Sección 4.8.1.

4.8. Diseño del motor de inferencia

El motor de inferencia es el corazón del módulo del ST: transforma un control legislativo en texto libre y un perfil de empresa en una **lista cerrada de requisitos técnicos a auditar**. Opera en las dos fases descritas en las Secciones 4.2 y 4.3 y resumidas en la Figura 4.6: una **Fase 1** de descubrimiento semántico, que localiza los controles del ENS pertinentes (Secciones 4.8.1 y 4.8.2), y una **Fase 2** que, para cada control descubierto,

Listado 4.1: Consulta Cypher de similitud coseno que recupera los controles candidatos (Fase 1).

```
MATCH (c:ControlENS)
WHERE c.vector IS NOT NULL
  AND NOT EXISTS { (c)-[:REFUERZA_A]->(:ControlENS) }
  // filtro por normativa del perfil (solo si ENS no aplica)
WITH c, gds.similarity.cosine(c.vector, $embedding_query) AS similarity
WHERE similarity > 0.65
ORDER BY similarity DESC
RETURN c.id_control, c.titulo, similarity
```

resuelve el Principio de Máximos directamente en *Cypher* (Sección 4.8.3).

4.8.1. Búsqueda por similitud

La Fase 1 arranca vectorizando el control legislativo (se combinan su título y su descripción en un único texto) con el mismo modelo `mxbai-embed-large` que se empleó para poblar el grafo, garantizando que consulta y nodos viven en el mismo espacio vectorial. Con ese vector, una consulta *Cypher* recupera los `ControlENS` candidatos (Listado 4.1).

Tres decisiones son relevantes. Primera, se **excluyen los controles que son refuerzos** (los que tienen una relación `REFUERZA_A` hacia otro control): un refuerzo no es un punto de entrada de la auditoría, sino algo que se inyecta cuando su control base resulta exigible (Sección 4.8.3). Segunda, el **umbral de similitud** (0,65) descarta el ruido de baja afinidad. Tercera, cuando el perfil de la empresa **no está sujeto al ENS**, la búsqueda **restringe los candidatos** a los controles que poseen relación `CUMPLE_DORA` y/o `CUMPLE_NIS2`, según las normativas que sí apliquen; cuando el ENS sí aplica, la búsqueda es universal sobre el catálogo. Esta restricción evita que la similitud semántica proponga controles que carecen de arista hacia la normativa del perfil y que, por tanto, la Fase 2 descartaría por falta de origen.

4.8.2. Juez LLM anti-falsos-positivos

La similitud vectorial es necesaria pero no suficiente: dos controles pueden compartir vocabulario de seguridad y evaluar prácticas distintas. Para depurar estos **falsos positivos**, los candidatos pasan por un **juez basado en un modelo de lenguaje** (clase `JuezControles`, sobre `qwen2.5:32b`). El juez recibe el control legislativo, el contexto de la empresa (sector, tamaño, categoría y dimensiones CITAD) y la lista numerada de candidatos, y devuelve en JSON el subconjunto que aportaría **evidencia directa** de cumplimiento, junto con una razón. El *prompt* le guía con criterios explícitos de inclusión y exclusión y le pide razonar, antes de decidir, qué medida

Cuadro 4.4: Pesos de exigencia empleados por el Principio de Máximos.

Nivel	Peso
`NINGUNO`	0
`BÁSICA` / `BAJO`	1
`MEDIA` / `MEDIO`	2
`ALTA` / `ALTO`	3

técnica exige el artículo y qué evidencia la demostraría.

Una característica de diseño es que el juez **no devuelve un número fijo** de controles, sino los que considera pertinentes. Para acotar su comportamiento, el sistema aplica varias salvaguardas: si hay dos candidatos o menos se omite el juez, se impone un **tope de seguridad** (10 controles) y, ante cualquier fallo (respuesta no parseable, error del modelo o ningún identificador válido), el sistema se queda con los tres candidatos de mayor similitud. La salida de la Fase 1 es, así, una lista depurada de controles base realmente pertinentes.

4.8.3. Principio de Máximos en *Cypher*

Para cada control base, la Fase 2 resuelve **qué requisitos auditar y a qué nivel de exigencia**, integrando ENS, NIS2 y DORA en una **única consulta *Cypher*** (`_buscar_requisitos_en_st`); su versión íntegra se reproduce en el Anexo A. El cálculo se apoya en un diccionario de pesos que ordena los niveles (Tabla 4.4), de modo que la regla del máximo se reduce a comparar enteros.

Validez de cada normativa

La consulta localiza el control y sus relaciones `CUMPLE_NIS2` y `CUMPLE_DORA`. Antes de calcular nada, determina si cada normativa **aplica de verdad**: la NIS2 se descarta si la relación marca `excluido_pce_nis2`, si la entidad es esencial/importante y el control está excluido para su tipo, o si la entidad no está sujeta a NIS2; DORA solo cuenta si la entidad es financiera y existe la relación. Esto evita arrastrar exigencias de normativas que no son de aplicación.

Nivel ganador

A continuación se calculan tres pesos y se toma el máximo (Listado 4.2).

El peso de la empresa se calcula **siempre** a partir de las dimensiones CITAD, con independencia de si el ENS es la normativa de aplicación legal.

Listado 4.2: Cálculo del nivel ganador como máximo de los tres pesos (Fase 2).

```
WITH c, nis2_valido, dora_valido,
     $pesos[ <suelo global NIS2, segun esencial/importante> ]    AS p_global,
     $pesos[ <suelo dimensional NIS2, segun esencial/importante>] AS p_dim,
     reduce(m = 0, dd IN c.dimensiones |
            CASE WHEN $pesos[$dims_empresa[dd]] > m
                 THEN $pesos[$dims_empresa[dd]] ELSE m END)      AS p_empresa
WITH c, nis2_valido, dora_valido,
     (CASE
        WHEN p_global >= p_dim AND p_global >= p_empresa THEN p_global
        WHEN p_dim   >= p_global AND p_dim   >= p_empresa THEN p_dim
        ELSE p_empresa
      END) AS peso_ganador
```

El parámetro `$dims_empresa` transporta el nivel de la organización en cada dimensión de seguridad CITAD, obtenido en la clasificación (Sección 4.11), y la expresión `reduce` toma el máximo de esos niveles *restringido a las dimensiones que el control afecta* (propiedad `dimensiones` del nodo). La razón de este desacoplamiento es conceptual: el CITAD mide el **riesgo real** de la organización (qué impacto tendría un incidente en cada dimensión), mientras que la aplicabilidad del ENS es una cuestión **jurídica** (si el régimen legal del ENS aplica a la entidad). Una empresa solo sujeta a DORA sigue teniendo un nivel de riesgo en sus dimensiones, y ese nivel debe graduar la profundidad de los requisitos técnicos del ENS que DORA reutiliza como sustrato. Los dos pesos de la NIS2 proceden de los suelos mínimos que la relación `CUMPLE_NIS2` almacena, ya en su variante global (`esencial_min_global` / `importante_min_global`) ya dimensional (`esencial_min_nivel` / `importante_min_nivel`), seleccionando la columna según el tipo de entidad. El nivel al que se auditará el control es el **máximo de esos tres pesos**: el del riesgo propio de la empresa (`p_empresa`), el suelo global de la NIS2 (`p_global`) y su suelo dimensional (`p_dim`). El flag `$aplica_ens` no interviene en este cálculo; su función se limita a gobernar la **atribución del origen** "ENS Base" o "ENS Ref" en los requisitos resultantes, de modo que un perfil sin ENS legal recibe requisitos graduados por CITAD pero etiquetados solo con sus orígenes DORA y/o NIS2.

Ejemplo de aplicación

La Tabla 4.5 muestra el cálculo sobre dos controles reales del grafo, para una entidad **esencial** sujeta al ENS (sin DORA). En el **caso A** (`Mp.com.1`), que afecta a las cinco dimensiones (por lo que su peso de empresa coincide con la categoría global ALTA), el suelo de la NIS2 es solo BÁSICA: gana el riesgo propio de la empresa. En el **caso B** (`Mp.com.2`), que afecta únicamente a la confidencialidad, una empresa con todas sus dimensiones en nivel Bajo

Cuadro 4.5: Ejemplo del Principio de Máximos: nivel ganador en dos controles reales (entidad esencial, ENS, sin DORA).

Caso	Control	Cat. empresa	Empresa	NIS2 glob.	NIS2 dim.	Ganador
A	`Mp.com.1`	ALTA	3	1	0	3 (ALTA) — riesgo propio
B	`Mp.com.2`	BÁSICA	1	0	3	3 (ALTA) — suelo NIS2

se ve **arrastrada hasta ALTA** por el suelo dimensional que la NIS2 impone a ese control (`esencial_min_nivel` = *Alto*). En ambos, peso_ganador = máx($p_{\text{emp}}, p_{\text{glob}}, p_{\text{dim}}$).

Requisitos base, refuerzos y consolidación

Fijado el nivel ganador, la consulta selecciona los `RequisitoENS` del control (`EVALUA_MEDIANTE`) cuyo umbral de `categorias_aplicables` queda alcanzado por ese nivel, y a cada uno le anota sus **orígenes** normativos (ENS, NIS2, DORA), conservando solo los que tengan al menos uno. Después **inyecta los refuerzos**: controles enlazados por `REFUERZA_A` que se activan por el nivel ganador o porque la NIS2 (`esencial_req` / `importante_req`) o DORA (`requiere_refuerzos`) los exijan, descartando los que la NIS2 excluya (`importante_excl`).

El detalle de cómo se han extraído y modelado todas estas reglas normativas desde las fuentes oficiales se recoge en el Anexo C. Finalmente, los requisitos base y los de refuerzo se **consolidan y deduplican** por descripción, fusionando los orígenes cuando un mismo requisito procede de varias vías, y producen la lista cerrada que el agente auditará.

Fundamento normativo de la evaluación por dimensión

Esta forma de ponderar el riesgo propio no es una elección de diseño arbitraria, sino la mecánica que prescribe el propio ENS. El Anexo II del RD 311/2022 emplea una doble escala sobre su tabla de medidas: las medidas marcadas "por categoría" se exigen según la categoría global del sistema (BÁSICA/MEDIA/ALTA), mientras que las marcadas con dimensiones concretas (D, TA, CITA...) se exigen según el nivel de esa(s) dimensión(es) (BAJO/MEDIO/ALTO); su apartado 2.1.e ordena seleccionar las medidas "de acuerdo con las dimensiones y sus niveles de seguridad y, para determinadas medidas de seguridad, de acuerdo con la categoría de seguridad del sistema".

El PCE-NIS2 hereda exactamente la misma convención: su tabla de aplicabilidad refleja la categoría cuando la medida afecta a las cinco dimensiones y el nivel dimensional cuando no [3] (tabla del apartado 6.2).

El cálculo por dimensión garantiza además dos propiedades: para los controles que afectan a las cinco dimensiones el resultado **coincide con la categoría global** (que el módulo CITAD deriva como máximo de los niveles dimensionales, Sección 4.11), y para los controles de dimensiones parciales evita exigir un control (por ejemplo, de pura disponibilidad) al nivel de una categoría global inflada por otra dimensión que ese control no toca. Si un perfil llega sin niveles dimensionales, cada dimensión hereda la categoría global y el cálculo se reduce de forma natural a la ponderación por categoría.

Operación detallada del motor

La Figura 4.6 hace explícita la **lógica de operación** del motor, con sus puntos de decisión reales: las dos salvaguardas del juez en la Fase 1 (se omite si hay dos candidatos o menos y, ante una salida no válida, se recurre a los tres controles de mayor similitud) y, en la Fase 2, la comprobación de validez de NIS2 y DORA previa al cálculo del nivel ganador. El Anexo E extiende esta vista con el flujo completo del módulo, incluyendo la *caché* de veredictos, la auditoría conversacional y la emisión del veredicto final.

4.9. Diseño del agente conversacional

Resuelta por el motor de inferencia la lista de requisitos a auditar (Sección 4.8), el **agente conversacional** se encarga de la interacción con el usuario: formula las preguntas en lenguaje de negocio, recoge las respuestas, emite un veredicto por control y agrega el resultado. A diferencia del juez, que emplea el modelo grande, el agente opera con un modelo más ligero (`qwen2.5:14b`), suficiente para la conversación. La Figura 4.7 resume su funcionamiento por turnos.

El primer turno es de **preparación**: ejecuta las dos fases del motor y construye la lista de requisitos, **deduplicando los controles base**; si el juez devolvió un refuerzo como `Op.pl.2.r2`, se evalúa su base `Op.pl.2`, pues la consulta *Cypher* ya extrae sus refuerzos aplicables. A partir de ahí, el agente recorre los requisitos **uno por turno**; completados todos los de un control emite su veredicto, pasa al siguiente y, al agotarlos, produce la evaluación final.

4.9.1. Generación de preguntas en lenguaje de negocio

Cada requisito técnico se traduce a una pregunta comprensible mediante el LLM, bajo el rol de un **analista de ciberseguridad**. El *prompt* le impone reglas estrictas: responder en español, sin jerga técnica ni siglas, di-

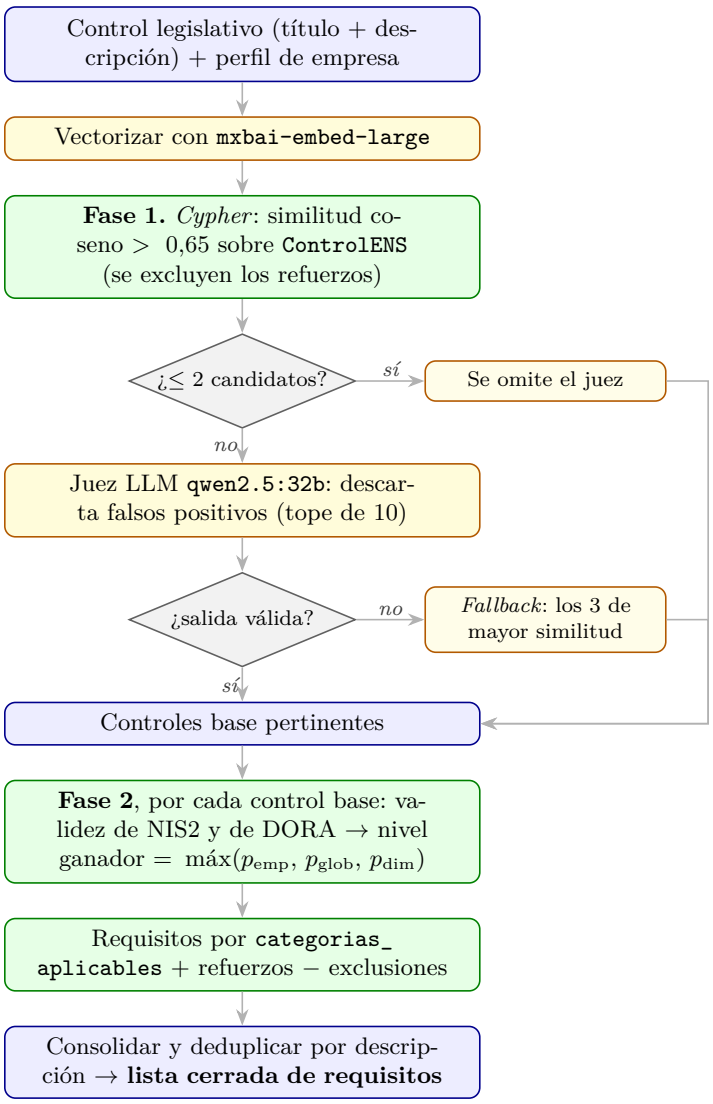

Figura 4.6: Operación detallada del motor de inferencia, con sus puntos de decisión. En la Fase 1, el juez LLM se omite si hay dos candidatos o menos y, ante una salida no válida, se recurre a los tres controles de mayor similitud; en la Fase 2, antes de calcular el nivel ganador se comprueba la validez de NIS2 y DORA para la entidad. El Anexo E extiende esta vista con el flujo completo del módulo del ST.

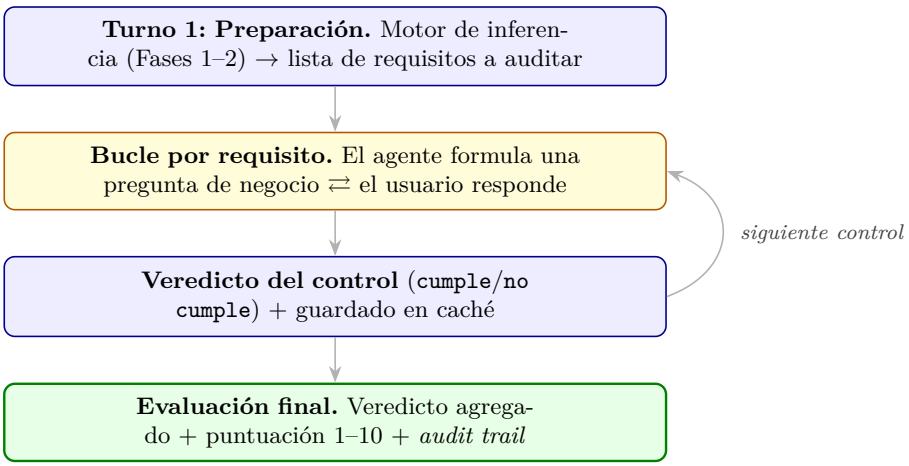

Figura 4.7: Flujo del agente conversacional por turnos: preparación, bucle de un requisito por turno, veredicto por control (con *caché*) y evaluación final con puntuación y traza de auditoría.

rigirse "vuestra empresaz, de forma decisiva, **aclarar la diferencia entre hacer algo de manera informal y cumplir realmente**. Cada pregunta termina con un párrafo en cursiva que comienza por “Para cumplir este punto necesitaríais:” seguido de dos o tres ejemplos concretos de la evidencia esperada.

La formulación se adapta al **tipo** de requisito (Sección 4.4): si es de **Evidencia**, la pregunta subraya que debe existir algo documentado o registrado de forma tangible; si es de **Verificación**, explora si la práctica se realiza y con qué frecuencia o alcance. Los requisitos **esenciales** se marcan explícitamente como críticos. Además, cada mensaje antepone un indicador de progreso (*Requisito i/N del control X*) que orienta al usuario sin recargar la conversación.

4.9.2. Evaluación por control y veredicto

El agente espera a haber recabado la información de todos los requisitos de un control y entonces emite un único veredicto para él. El LLM recibe el historial completo de la conversación y la lista de requisitos, con su tipo y marca de esencial, y devuelve un JSON `{cumple, razon}`. Para garantizar que la respuesta del modelo sea siempre parseable, la llamada de evaluación emplea **salida estructurada** (*structured output*): se fija un *JSON Schema* que exige las claves `cumple` (booleano) y `razon` (cadena), y el motor Ollama garantiza una salida JSON sintácticamente válida y con ambas claves siempre presentes. Este mecanismo se aplica únicamente a la llamada de evaluación; la generación de preguntas en lenguaje de negocio sigue siendo

Cuadro 4.6: Puntuación determinista de cumplimiento (1–10).

Condición	**Puntuación**
Con algún incumplimiento esencial:	
$> 70\,\%$ de controles fallidos	1
$> 40\,\%$ de controles fallidos	2
resto	3
Sin incumplimiento esencial (según % de controles que cumplen):	
$100\,\%$	10
$\geq 90\,\%$	9
$\geq 75\,\%$	8
$\geq 60\,\%$	7
$\geq 50\,\%$	6
$\geq 35\,\%$	5
$< 35\,\%$	4

texto libre.

El criterio del veredicto se calibra por tipo y con una tolerancia explícita: para los requisitos de **Evidencia** basta cualquier rastro documental (una hoja de cálculo, un *ticket*, un correo archivado), reservando el "no cumple" para la ausencia total; los de **Verificación** se dan por cumplidos si la empresa describe de forma creíble cómo realiza la práctica. Sobre todo ello prevalece la **regla esencial**: si un requisito marcado como esencial no se cumple, el control entero se considera no cumplido, con independencia del resto.

4.9.3. Puntuación determinista 1–10 y *caché* de veredictos de sesión

Puntuación

Aunque el veredicto de cada control lo emite el LLM, la **puntuación global (1–10) es completamente determinista** (`_calcular_puntuacion`): es una función del porcentaje de controles que cumplen y de si alguno con requisitos esenciales ha fallado (Tabla 4.6). La presencia de un incumplimiento esencial confina la nota al tramo bajo (1–3); en su ausencia, la nota crece con el porcentaje de controles cumplidos (4–10).

Caché de veredictos.

Para no preguntar dos veces por el mismo control técnico, el agente mantiene una *caché* de sesión (`cache_veredictos`) indexada por el identificador del control base, que guarda su veredicto, la razón y el control legislativo que lo originó. La *caché* persiste durante toda la sesión: el método `reset`,

que prepara el agente para el siguiente control legislativo, reinicia los índices y las respuestas pero conserva la *caché*, de modo que un control técnico ya evaluado se reaprovecha si reaparece bajo otro artículo. Un detalle de ordenación es relevante: la comprobación de **aplicabilidad al perfil** (si el control tiene requisitos para la empresa) se realiza **antes** de consultar la *caché*; así, un veredicto cacheado no se reutiliza para un perfil en el que ese control carezca de requisitos aplicables.

Cuando un control legislativo no produce ningún candidato por encima del umbral de similitud, o ninguno de los candidatos genera requisitos aplicables al perfil, el módulo del ST devuelve una señal de **salto** (`sin_control_tecnico`), sin emitir veredicto ni mostrar mensaje al usuario. El Coordinador avanza al siguiente control y el artículo omitido queda registrado en la traza de auditoría con `cumple=None`, separándolo de los evaluados en el informe final para no producir un falso "cumple".

4.9.4. *Audit trail* por sesión

Al cerrar la evaluación de cada control legislativo, el agente vuelca una **traza de auditoría** (`_guardar_audit_trail`): un **único fichero JSON por sesión** que se amplía con cada control. Cada entrada registra el control legislativo evaluado, el **resultado del ST** (cumple, puntuación 1–10 y razón) y el detalle de los **controles técnicos** auditados: identificador, veredicto, razón del LLM y similitud que motivó su selección. La frontera de responsabilidades es la ya descrita: el módulo del ST aporta el veredicto técnico de cada control legislativo y el **veredicto global lo emite el SN**. Esta traza es la que se reconstruye en la Figura 4.1 de la Sección 4.5.

4.10. Operación interna del módulo del ST en una evaluación

Las secciones anteriores describieron por separado el motor de inferencia (Sección 4.8) y el agente conversacional (Sección 4.9). La Figura 4.8 los integra **en el tiempo**: muestra la operación interna del módulo del ST al procesar un control, con sus llamadas reales a **Neo4j** (*Cypher*) y a **Ollama** (*embeddings*, juez y agente).

4.11. Módulo CITAD

Toda la evaluación técnica parte de un dato: la **categoría ENS** de la empresa y el nivel de cada una de sus cinco dimensiones de seguridad

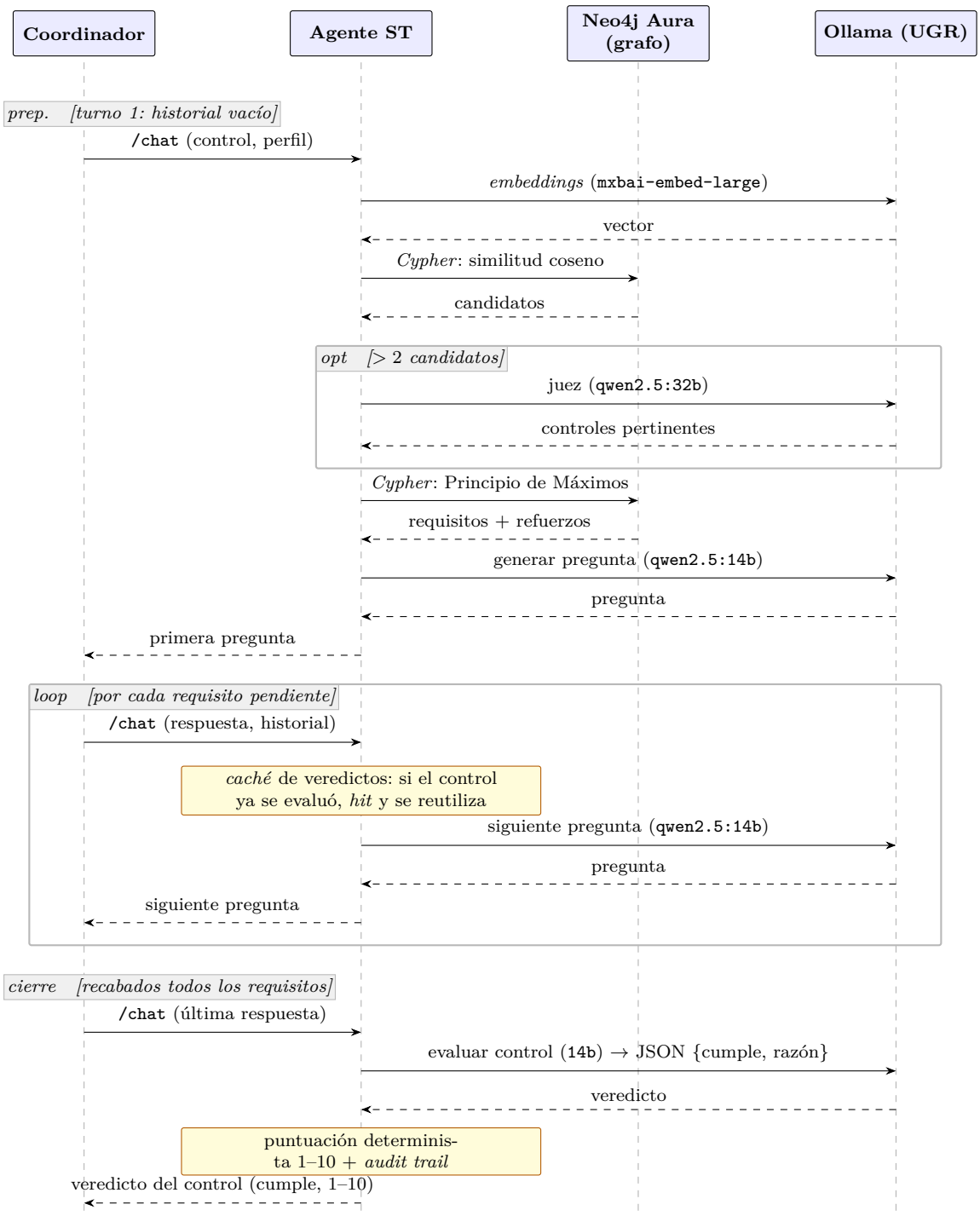

Figura 4.8: Diagrama de secuencia de la operación interna del módulo del ST al procesar un control, con sus llamadas a Neo4j (*Cypher*) y a Ollama (*embeddings* `mxbai`, juez `qwen2.5:32b` y agente `qwen2.5:14b`). El turno de preparación ejecuta las dos fases del motor; el bucle recorre los requisitos uno por turno, reutilizando la *caché* ante un control ya evaluado; el cierre emite el veredicto y la puntuación determinista. Es el detalle interno de la línea de vida del ST que el Anexo D trata como caja con nota.

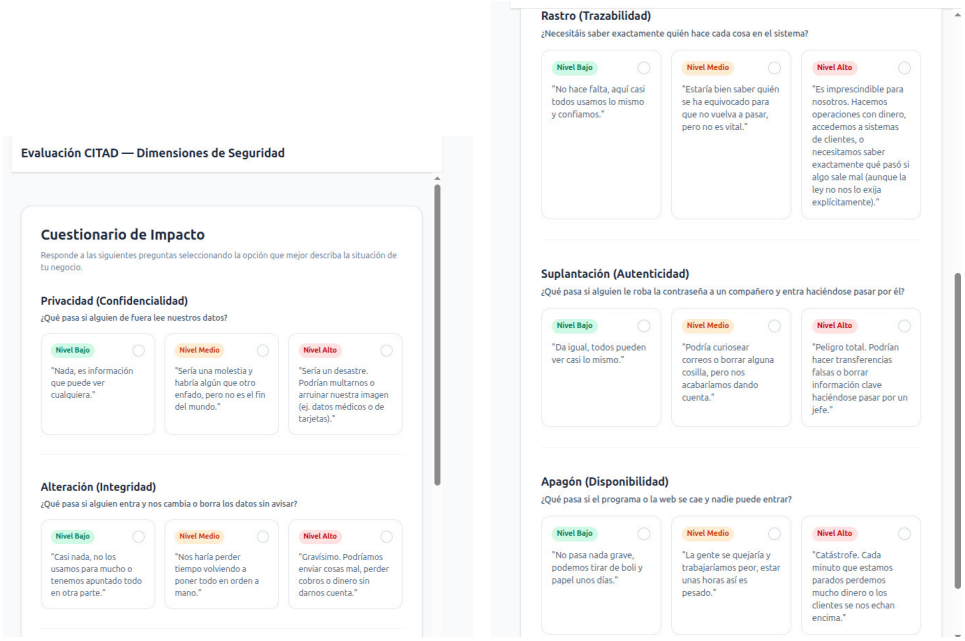

Figura 4.9: Cuestionario de impacto CITAD en el *frontend*: una pregunta por dimensión, redactada como escenario de negocio, con tres respuestas que corresponden a nivel Bajo, Medio o Alto.

(Sección 3.2.3). Obtener ese dato es la función del componente **CITAD**, un cuestionario de impacto de **autoría propia** (Capítulo 1) integrado en el *frontend*.

CITAD plantea **una pregunta por dimensión**, redactada como un escenario de negocio (por ejemplo, "¿qué pasa si el programa o la web se cae y nadie puede entrar?" para la Disponibilidad), con tres respuestas que corresponden a un nivel **Bajo**, **Medio** o **Alto**. A partir de las cinco respuestas, el componente aplica una lógica **completamente determinista, sin intervención de ningún modelo de lenguaje** (Figura 4.10): cada respuesta fija el nivel de su dimensión (BÁSICA, MEDIA o ALTA) y la **categoría global** se obtiene por la **regla del máximo** (la categoría es la más alta de las cinco dimensiones), en coherencia con el Anexo I del ENS. La Figura 4.9 muestra el cuestionario tal como lo ve el usuario en el *frontend*.

El resultado, la categoría global y el nivel de cada dimensión, se incorpora al perfil de la empresa y es la entrada que parametriza el Principio de Máximos descrito en la Sección 4.8.3. La Figura 4.11 muestra ese resultado para un perfil de ejemplo con dimensiones mixtas.

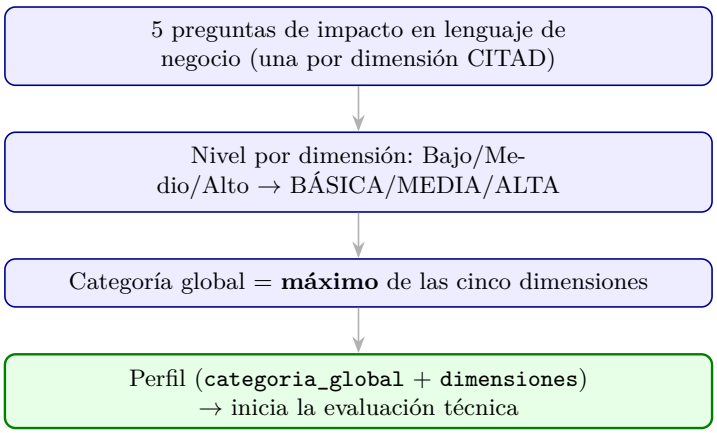

Figura 4.10: Lógica determinista del componente CITAD: cinco preguntas de impacto (una por dimensión) determinan el nivel de cada dimensión, y la categoría global se obtiene por la regla del máximo.

Figura 4.11: Resultado del componente CITAD para un perfil de ejemplo: el nivel de cada dimensión (C y A en MEDIA; I, T y D en BÁSICA) determina, por la regla del máximo, una categoría global MEDIA, que da comienzo a la evaluación técnica.

Cuadro 4.7: Contratos de interfaz entre los servicios.

Llamada	***Endpoint***	**Carga principal → respuesta**
Frontend → Coord.	`POST /session/start`	`profile` → `session_id`, normativas, primera pregunta
Frontend → Coord.	`POST /session/message`	`session_id`, `message` → mensaje del agente, progreso, `finished`, veredicto final
Coord. → SN	`POST /controls`	`profile` → `controls[]`, `normativas[]`
Coord. → ST	`POST /chat`	`session_id`, `control`, `message`, `history`, `profile` → `response`, `finished`, `cumple`, `puntuacion`
Coord. → SN	`POST /evaluate`	`session_id`, `profile`, `results` → cumplimiento, porcentaje, informe

4.12. Integración con el Sistema Normativo (SN) y el Coordinador

El módulo del ST no funciona de forma aislada: se integra con el resto de *Ciber-AsesorIA* a través de una **capa de comunicación vía API**, desarrollada también como parte de este trabajo (Capítulo 1). El diagrama de secuencia del Anexo D (Figura D.1) muestra el recorrido completo de una evaluación de extremo a extremo, y la Tabla 4.7 resume los contratos de interfaz.

El Coordinador como orquestador

El Coordinador (FastAPI, `:5000`) mantiene el **estado de sesión en memoria**: un diccionario indexado por `session_id` con el perfil, la lista de controles, el índice actual, el historial y los resultados. Por cada control legislativo abre una conversación con el módulo del ST y, cuando este marca el control como terminado, guarda su veredicto y avanza al siguiente; agotados todos, envía el conjunto de resultados al SN, que emite el veredicto narrativo final. El *frontend* nunca habla directamente con el *backend*: un **proxy** en Next.js (`/api/chat`) reenvía las peticiones al Coordinador, evitando problemas de CORS y ocultando la URL interna.

Gestión de la conversación multi-turno

Cada llamada a `/chat` entrega al módulo del ST el control, el perfil y el **historial completo** de la conversación. El agente **mantiene en memoria el estado del control en curso** (turno actual, requisito en evaluación, veredictos parciales); para delimitar dónde empieza cada control, el servicio interpreta un **historial vacío como el inicio de uno nuevo** y, en ese caso,

reinicia el agente (`reset`) y fija el identificador de sesión para la traza de auditoría. Los turnos sucesivos avanzan requisito a requisito (Sección 4.9).

Normalización de claves (*frontend* ↔ *Cypher*).

Un detalle de integración recurrente es que el *frontend* nombra los campos en inglés (`appliesDORA`, `nis2Type`) mientras la lógica interna y las consultas *Cypher* esperan otra nomenclatura. La conversión se resuelve en dos puntos: los modelos `Pydantic` declaran **alias** que aceptan ambas formas, y el motor de inferencia **normaliza el perfil** antes de la consulta (por ejemplo, traduce `nis2Type = "None"` a `"NINGUNO"` y deriva `es_dora`/`aplica_ens` de los campos de aplicabilidad), de modo que el *Cypher* del Principio de Máximos recibe siempre parámetros consistentes.

Resiliencia

Por último, dado que la fase de descubrimiento emplea el modelo de lenguaje de mayor tamaño (más lento), el Coordinador fija un **tiempo de espera amplio** (200 segundos) en sus llamadas al módulo del ST, en línea con el comportamiento ante fallos descrito en la Sección 4.5.

Capítulo 5

Implementación

Este capítulo aborda el desarrollo práctico del módulo del Sistema Técnico. A partir de la arquitectura definida previamente, aquí se detalla el conjunto de tecnologías escogidas para su construcción y se analizan las piezas de código clave que trasladan a la práctica la lógica del motor de inferencia, las consultas al grafo normativo y la integración de los modelos de lenguaje.

5.1. Tecnologías empleadas

El módulo del ST se ha construido priorizando el uso de ***software* y modelos de pesos abiertos**, con el fin de minimizar la dependencia de servicios comerciales de pago. La Tabla 5.1 resume las piezas principales y sus versiones; las decisiones estratégicas de arquitectura, como el apoyo en la infraestructura institucional de la UGR para la inferencia, se justifican a continuación.

FastAPI como base de los servicios

Tanto el servicio ST como el Coordinador se exponen con **FastAPI** sobre el servidor ASGI **Uvicorn**. La elección responde a que las llamadas entre servicios son operaciones de E/S de larga duración (la inferencia de los modelos puede tardar decenas de segundos): el modelo asíncrono de FastAPI permite mantener esas llamadas sin bloquear el proceso, y su integración nativa con **Pydantic** aporta validación automática de los contratos de API descritos en la Sección 4.12.

Cuadro 5.1: Tecnologías principales del módulo del Sistema Técnico y la capa de conexión.

Ámbito	**Tecnología (versión)**	**Función**
Lenguaje	Python 3.12	*Backend* del módulo del ST y la capa de conexión
API REST	FastAPI 0.115 + Uvicorn 0.34	Servicios asíncronos ST y Coordinador
Validación	Pydantic 2.11	Modelos de entrada/salida y *alias* de campos
Cliente HTTP	httpx 0.28	Llamadas asíncronas entre servicios
Grafo	Neo4j (*driver*) + GDS	Grafo de conocimiento y similitud vectorial
Capa LLM	LangChain (`langchain-ollama`)	Abstracción sobre el modelo y los *embeddings*
Modelos	qwen2.5:14b / 32b, mxbai-embed-large	Agente, juez y *embeddings* (vía Ollama)
Config.	python-dotenv	Carga de credenciales y parámetros desde `.env`
Frontend (contexto, desarrollado como parte del conjunto)		
UI	Next.js 16 + React 19	Wizard de perfilado y chat guiado
Estado	Zustand 5	Estado global del cliente
Estilos / UI	Tailwind 4, Radix UI, Recharts	Componentes, estilos y gráficos

LangChain como capa sobre el LLM local

El acceso a los modelos no se hace contra la API HTTP de Ollama directamente, sino a través de la abstracción `langchain-ollama` (clases `ChatOllama` y `OllamaEmbeddings`). Esto desacopla el código del proveedor concreto (cambiar de modelo o de *backend* se reduce a una variable de entorno) y unifica el patrón de invocación (Sección 5.2.1). El *driver* oficial de **Neo4j** y su librería *Graph Data Science* (GDS) completan el acceso al grafo.

Infraestructura LLM de coste cero

El grueso del sistema (la lógica del agente, el motor de inferencia y los servicios FastAPI del módulo del ST y el Coordinador) se ejecuta **en local**, en el equipo de desarrollo. Solo dos dependencias residen fuera de la máquina: el **grafo de conocimiento**, en *Neo4j Aura* (nube), y los **modelos de lenguaje**. Para estos últimos, una decisión de implementación con impacto directo en la viabilidad económica del proyecto es no ejecutarlos en *hardware* propio ni en servicios de pago, sino en una **instancia de Ollama alojada en la Universidad de Granada**: el código apunta siempre a `http://localhost:11435`, pero ese puerto local es en realidad la **boca de un túnel SSH** hacia la máquina de la universidad, donde residen `qwen2.5:14b` (agente), `qwen2.5:32b` (juez) y `mxbai-embed-large` (*embeddings*). La Figura 5.1 ilustra este despliegue real, complementario a la arquitectura lógica de la Figura 4.2: todo corre en local salvo el grafo (Aura) y la inferencia (UGR vía túnel SSH). Externalizar la inferencia a un servidor institucional sin coste y apoyar el grafo en la capa gratuita de *Neo4j Aura* permite usar modelos de 14.000 y 32.000 millones de parámetros con **coste de inferencia nulo** para el proyecto.

Selección de modelos

La elección estuvo condicionada por el catálogo de modelos ya instalados en el servidor (Tabla 5.2). Para el agente y el juez se adoptó la familia de pesos abiertos `qwen2.5` [28]: el modelo de 14.000 millones de parámetros para el agente conversacional y el de 32.000 millones para el juez, que requiere mayor capacidad de razonamiento para descartar falsos positivos (Sección 4.8.2). Para los *embeddings*, la oferta se reducía a dos modelos; se escogió `mxbai-embed-large` por su mayor precisión en preguntas largas y contextuales, el escenario más próximo al de este trabajo, que vectoriza nodos extensos con sus notas aclaratorias fusionadas, según una comparativa independiente de modelos abiertos para tareas de recuperación [25].

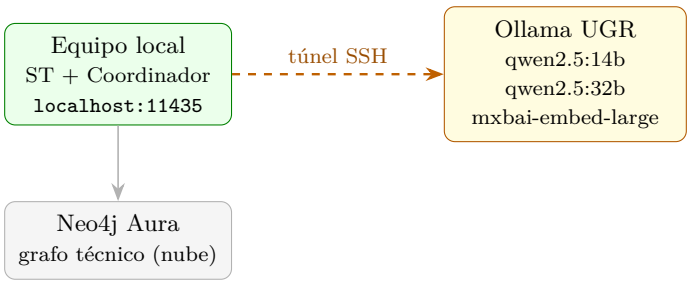

Figura 5.1: Despliegue real de la inferencia: el módulo del ST direcciona los modelos como `localhost:11435`, que es la boca de un túnel SSH hacia la instancia de Ollama de la Universidad de Granada. El grafo técnico reside en *Neo4j Aura*. Este esquema concreta, a nivel de despliegue, la arquitectura lógica de la Figura 4.2.

Cuadro 5.2: Catálogo de modelos disponibles en el servidor de la UGR (parámetros según la etiqueta de Ollama). La oferta de modelos de *embeddings* se reducía a dos. En negrita, los seleccionados para el módulo del ST.

Modelo	**Familia**	**Parám.**	**Uso / observación**
Modelos de embeddings (únicos disponibles)			
`mxbai-embed-large`	Mixedbread	334 M	**Elegido** (*embeddings*)
`nomic-embed-text`	Nomic	137 M	—
Modelos de lenguaje generalistas			
`qwen2.5:32b`	Qwen2.5	32.800 M	**Elegido** (juez)
`qwen2.5:14b`	Qwen2.5	14.800 M	**Elegido** (agente)
`qwen2.5:7b`	Qwen2.5	7.600 M	—
`deepseek-r1:32b`	DeepSeek	32.800 M	—
`phi4`	Phi-4	14.700 M	—
`phi4-reasoning`	Phi-4	14.700 M	—
`gpt-oss:20b`	gpt-oss	20.900 M	—
`mistral`	Mistral	7.200 M	—
`llama3`	Llama	8.000 M	—
`llama3.2`	Llama	3.200 M	—
`llama2`	Llama	7.000 M	—
Modelos especializados			
`qwen2.5-coder`	Qwen2.5	7.600 M	Código
`deepseek-coder`	DeepSeek	1.000 M	Código
`llava:7b`	LLaVA	7.000 M	Multimodal (visión)

Listado 5.1: Inicialización de la capa LLM (agente, juez y embeddings).

```
model_name = os.getenv("ST_OLLAMA_MODEL", "qwen2.5:14b")
base_url   = os.getenv("ST_OLLAMA_HOST", "http://localhost:11435")

self.llm        = ChatOllama(model=model_name, base_url=base_url, temperature=0.0)
self.embeddings = OllamaEmbeddings(model="mxbai-embed-large", base_url=base_url)

# El juez puede usar un modelo mayor (qwen2.5:32b); si coincide, se reutiliza.
juez_model = os.getenv("JUEZ_OLLAMA_MODEL", model_name)
llm_juez   = (ChatOllama(model=juez_model, base_url=base_url, temperature=0.0)
              if juez_model != model_name else self.llm)
self.juez  = JuezControles(llm_juez)
```

5.2. Desarrollo del motor de inferencia

El motor de inferencia diseñado en la Sección 4.8 se materializa en la clase `AgenteEvaluacionControles` (fichero `st_agente_evaluacion.py`). Su constructor fija la configuración común de toda la inferencia: el modelo del agente, el del juez y el de *embeddings*, todos servidos por el mismo Ollama y todos con **temperatura 0**, porque el sistema actúa como auditor determinista y no como generador creativo.

Todas las invocaciones al modelo siguen el mismo patrón de **cadena de LangChain**: se construye un `ChatPromptTemplate` con un mensaje de sistema y uno de usuario, se compone con el modelo mediante el operador `|` y se invoca; la respuesta se lee de `respuesta.content`. Este patrón se repite en la generación de preguntas, la evaluación de controles y el juez.

5.2.1. Pipeline de *embeddings* y consulta vectorial

Vectorización previa

Los *embeddings* del grafo no se calculan en tiempo de consulta sino una sola vez, durante el poblado, mediante una cadena de tres etapas por normativa (*ingesta* → *vectorización* → *carga*), presentes en `ST/ENS`, `ST/NIS2` y `ST/DORA`. El vectorizador del ENS recorre el JSON jerárquico y, por cada control y requisito, llama a `embed_query` con el mismo modelo que después usará la consulta, fusionando las notas aclaratorias en el texto del requisito para enriquecer su representación semántica:

Consulta vectorial

En cada evaluación, la Fase 1 (Sección 4.8.1) vectoriza el control legislativo entrante y lanza contra Neo4j la consulta de similitud coseno ya presentada en el listado de la Sección 4.8.1. El Listado 5.3 muestra el *envol-*

Listado 5.2: Vectorización previa de un requisito, con fusión de notas (st_vectorizer_ens.py).

```
texto_hijo = req.get("descripcion", "")
notas = req.get("notas_adicionales", [])
if notas:
    texto_notas = " ".join(notas)
    texto_hijo = f"{texto_hijo}\nContexto adicional: {texto_notas}"
embedding_hijo = self._embed_text(texto_hijo)   # -> embed_query(mxbai-embed-large)
```

Listado 5.3: Envoltorio Python de la búsqueda por similitud (`_buscar_controles_por_similitud`).

```
texto_combinado = f"{titulo_normativo}. {texto_normativo}" if titulo_normativo else
    texto_normativo
embedding_query = self.embeddings.embed_query(texto_combinado)

driver = get_st_driver()   # driver unico compartido (vease mas abajo)
with driver.session() as session:
    # 'query' = el Cypher de similitud coseno (umbral 0.65) del cap. de Diseño
    result = session.run(query, embedding_query=embedding_query)
    for record in result:
        candidatos.append({"id": record["id_control"],
                           "titulo": record.get("titulo", record["id_control"]),
                           "similitud": record["similarity"]})

# Los candidatos pasan al juez LLM (Fase 1, anti-falsos-positivos)
seleccionados = self.juez.seleccionar(titulo_normativo, texto_normativo, normativa,
    perfil, candidatos)
```

torio Python: combinación de título y descripción, llamada a `embed_query`, apertura de una sesión sobre el *driver* compartido y recogida de los candidatos como diccionarios con su similitud. Por concisión, la variable `query` del listado no reproduce el *Cypher*: es exactamente la consulta de similitud coseno presentada en la Sección 4.8.1.

Driver de Neo4j como *singleton* de módulo.

El acceso al grafo se canaliza a través de un **único objeto *driver* compartido**, que se crea **la primera vez** que se necesita (`get_st_driver()`) y reutilizado por todas las consultas del módulo. El *driver* oficial de Neo4j está diseñado para ese uso: es un objeto de larga vida, seguro frente a hilos, que gestiona internamente un **pool de conexiones**; cada operación se limita a abrir una sesión ligera sobre él. La alternativa, instanciar un *driver* por consulta, obligaría a repetir el *handshake* TLS y la autenticación contra Aura en cada una de las decenas de consultas que componen una auditoría, multiplicando la latencia de la evaluación: en las baterías de validación del Capítulo 6, que encadenan cientos de ejecuciones del motor, la reutilización

Listado 5.4: Driver Neo4j compartido, creado en la primera consulta y reutilizado después.

```
_ST_DRIVER = None

def get_st_driver():
    """Devuelve el driver Neo4j compartido del módulo del ST, creandolo solo la
     primera vez."""
    global _ST_DRIVER
    if _ST_DRIVER is None:
        if not NEO4J_URI_ST or not NEO4J_PASSWORD_ST:
            return None
        _ST_DRIVER = GraphDatabase.driver(
            NEO4J_URI_ST, auth=(NEO4J_USER_ST, NEO4J_PASSWORD_ST))
    return _ST_DRIVER
```

Listado 5.5: Diccionario de pesos del Principio de Máximos.

```
_PESOS = {"NINGUNO": 0, "BÁSICA": 1, "BAJO": 1, "MEDIA": 2, "MEDIO": 2, "ALTA": 3, "
     ALTO": 3}
```

del *driver* redujo el tiempo total de ejecución de varios minutos a menos de un minuto.

5.2.2. Implementación del Principio de Máximos

Para cada control base que devuelve la Fase 1, la **Fase 2** resuelve qué requisitos auditar y a qué nivel de exigencia: el **Principio de Máximos** (Sección 4.8.3) se implementa en la función `_buscar_requisitos_en_st`, que lo resuelve todo en una única consulta *Cypher* (reproducida completa en el Anexo A). La pieza que convierte la regla del máximo en una comparación de enteros es un diccionario de pesos, el mismo de la Tabla 4.4, que se pasa como parámetro a la consulta:

Normalización del perfil

Antes de ejecutar la consulta, el perfil que llega del *frontend* se normaliza para que el *Cypher* reciba siempre parámetros consistentes (Sección 4.12): la cadena `"None"` se traduce a `"NINGUNO"`, el tipo de entidad se pasa a mayúsculas y los *flags* de DORA y ENS se derivan aceptando tanto la forma en español como la inglesa. La misma normalización construye el diccionario `dims_empresa` con el nivel de la organización en cada dimensión CITAD, la entrada de la ponderación dimensional del riesgo propio (Sección 4.8.3): si el perfil trae los niveles del módulo CITAD se usan tal cual, y para cualquier dimensión ausente se hereda la categoría global, de modo que un perfil sin detalle dimensional reproduce exactamente la ponderación por categoría.

Listado 5.6: Normalización del perfil y parametrización de la consulta.

```
cat_global    = perfil.get("categoria_global", "BÁSICA")
tipo_nis2_raw = perfil.get("tipo_nis2") or perfil.get("nis2_type") or "NINGUNO"
tipo_nis2     = "NINGUNO" if tipo_nis2_raw in ("None", "none", None) else
    tipo_nis2_raw.upper()
es_dora       = perfil.get("es_dora",    perfil.get("applies_dora", False))
aplica_ens    = perfil.get("aplica_ens", perfil.get("applies_ens",  True))

# Niveles por dimension (CITAD); sin ellos, cada dimension hereda la categoria
    global
dims_in      = perfil.get("dimensiones") or {}
dims_empresa = {d: str(dims_in.get(d, dims_in.get(d.lower(), cat_global)) or
    cat_global)
                for d in ("C", "I", "T", "A", "D")}

result = session.run(query, id_control=id_control, cat_global=cat_global,
                     dims_empresa=dims_empresa, tipo_nis2=tipo_nis2,
                     es_dora=es_dora, aplica_ens=aplica_ens, pesos=self._PESOS)
```

Listado 5.7: Activación y exclusión de refuerzos (fragmento Cypher).

```
WHERE (
    (peso_ganador >= 2 AND "MEDIA" IN ref.categorias_aplicables)
    OR (peso_ganador >= 3 AND "ALTA"  IN ref.categorias_aplicables)
    OR ANY(r IN n_add WHERE toLower(ref.id_control) ENDS WITH "." + toLower(r))
    OR ANY(r IN d_add WHERE toLower(ref.id_control) ENDS WITH "." + toLower(r))
)
AND NOT ANY(excl IN n_del WHERE toLower(ref.id_control) ENDS WITH "." + toLower(excl
    ))
```

Inyección de refuerzos en *Cypher*. La parte más delicada de la consulta es la activación de refuerzos. Un refuerzo (control enlazado por `REFUERZA_A`) entra en la auditoría si el nivel ganador alcanza su categoría **o** si una normativa lo exige explícitamente (`n_add` para NIS2, `d_add` para DORA), y se descarta si la NIS2 lo perdona a las entidades *importantes* (`n_del`). Estas listas se obtienen de las propiedades del "grafo gordo" (Sección 4.7.2):

Consolidación y deduplicación

La consulta devuelve por separado los requisitos base y los de refuerzo; el código Python los **fusiona indexando por descripción**, de modo que un requisito que procede de varias vías (p. ej. ENS y NIS2) aparece una sola vez con sus orígenes combinados, tal como ilustra la Figura 5.2. Esto produce la lista cerrada que auditará el agente.

Listado 5.8: Consolidación de requisitos por descripción, combinando orígenes.

```
for r in reqs:
    desc = r.get("desc")
    if desc not in todos_requisitos:
        todos_requisitos[desc] = r
    else:                                   # mismo requisito por otra vía: fusionar
      orígenes
        existentes = set(todos_requisitos[desc].get("origenes", []))
        for org in r.get("origenes", []):
            if org not in existentes:
                todos_requisitos[desc]["origenes"].append(org)
                existentes.add(org)
```

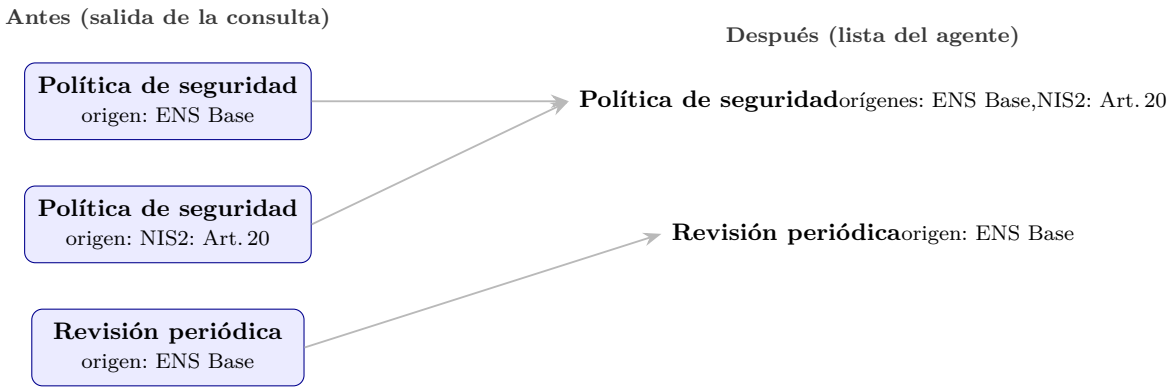

Figura 5.2: Consolidación de requisitos por descripción (ejemplo ilustrativo). Un mismo requisito puede llegar por varias vías (por ejemplo, una medida que el ENS exige por su categoría y que la NIS2 vuelve a imponer como suelo). Indexando por la descripción, el código lo deja una sola vez y **une sus etiquetas de origen**, produciendo la lista cerrada y sin duplicados que audita el agente, con la trazabilidad de cada requisito preservada.

Listado 5.9: *Prompt* de sistema para traducir un requisito a lenguaje de negocio (extracto).

```
Eres un analista de ciberseguridad especializado en ENS, NIS2 y DORA.
Tu tarea es convertir un requisito técnico normativo en una pregunta clara y
comprensible para una empresa (lenguaje de negocio).

REGLAS OBLIGATORIAS:
- RESPONDE SIEMPRE EN ESPAÑOL.
- NO uses jerga técnica, siglas legales ni el texto técnico crudo.
- Habla de "vuestra empresa", nunca de "el control" o "la norma".
- Si el usuario podría confundir "lo hacemos de forma informal" con "cumplimos", ACLÁRALO.
- Termina SIEMPRE con un párrafo que empiece por "Para cumplir este punto necesitaréis:"
  seguido de 2-3 ejemplos concretos de lo que se esperaría encontrar.
```

5.3. Desarrollo del agente conversacional

El agente conversacional (Sección 4.9) se implementa sobre el mismo patrón de cadenas de LangChain, con tres responsabilidades en código: **generar** las preguntas, **evaluar** cada control y **extraer** de forma robusta el JSON que devuelve el modelo.

Generación de preguntas

El *prompt* de sistema impone las reglas de redacción descritas en la Sección 4.9.1 (español, sin jerga, dirigirse a "vuestra empresa", cerrar con un párrafo de pista) y se adapta dinámicamente según el requisito sea de **Evidencia** o **Verificación** y según esté marcado como **esencial**.

Evaluación de un control técnico

Tras recabar las respuestas de todos los requisitos de un **control técnico**, el agente emite un **único veredicto binario** (`cumple` / `no cumple`) para él. La puntuación numérica (1–10) no se decide en este punto: corresponde al **veredicto del control legislativo** y se obtiene de forma determinista al agregar todos sus controles técnicos (apartado "Puntuación determinista" de esta misma sección). El *prompt* de sistema fija la calibración por tipo y la regla esencial (Sección 4.9.2).

Para garantizar que la respuesta cumpla el contrato esperado, el agente no confía únicamente en las instrucciones del *prompt*, sino que emplea la **salida estructurada** (*structured output*): se define un *JSON Schema* (`formato_veredicto`) y se vincula al modelo mediante `bind(format=formato_veredicto)`. Esto obliga al motor de inferencia a devolver siempre un objeto

Listado 5.10: Uso de JSON Schema para garantizar la estructura del veredicto.

```
formato_veredicto = {
    "type": "object",
    "properties": {
        "cumple": {"type": "boolean"},
        "razon": {"type": "string"},
    },
    "required": ["cumple", "razon"],
}
chain = prompt | self.llm.bind(format=formato_veredicto)
respuesta = chain.invoke(...)
```

JSON válido con las claves booleanas y de texto requeridas.

Extracción robusta de JSON como salvaguarda

Aunque la salida estructurada garantiza la sintaxis y las claves, la respuesta cruda pasa por un *helper* `_extraer_json` que la normaliza en **tres pasadas**. La primera intenta parsear el texto directamente con `json.loads`: como el esquema obliga al modelo a devolver JSON puro, este es el caso habitual y el más fiable, pues evita que una llave `}` contenida en el campo `razon` confunda a una expresión regular. Solo si el modelo envuelve el JSON en un bloque de código Markdown (las comillas triples "`` `json ``) o lo acompaña de texto adicional entran las dos pasadas de respaldo basadas en expresiones regulares: busca el bloque Markdown y, si no lo encuentra, cualquier objeto entre llaves. Ante cualquier fallo inesperado devuelve `None`, lo que asegura una degradación elegante a un veredicto conservador sin tumbar la evaluación.

Control de turnos y *caché*. El estado de la conversación (índices de control y de requisito, respuestas y veredictos parciales) vive en atributos de la instancia. El método `reset`, invocado al empezar cada control legislativo, reinicia ese estado pero **conserva deliberadamente la *caché* de veredictos** (Sección 4.9.3), de modo que un control técnico ya evaluado no se vuelve a preguntar si reaparece bajo otro artículo:

Puntuación determinista

En coherencia con la frontera diseñada (Sección 4.9.3), la nota 1–10 no la asigna el modelo de lenguaje, sino que se deriva mediante una regla fija a partir del porcentaje de controles que cumplen y de si ha fallado alguno con requisitos esenciales (`_calcular_puntuacion`), según la Tabla 4.6. Un fallo esencial confina la nota al tramo 1–3; en su ausencia, crece con el porcentaje de cumplimiento hasta 10.

Listado 5.11: Extracción del objeto JSON en tres pasadas (directa, bloque Markdown y fallback de llaves).

```
def _extraer_json(self, texto):
    if not texto:
        return None
    # 0) Directa: con format=<schema> el modelo devuelve JSON puro
    try:    return json.loads(texto.strip())
    except Exception: pass
    match = re.search(r'```json\s*(\{.*?\})\s*```', texto, re.DOTALL)   # 1) bloque
     markdown
    if match:
        try:    return json.loads(match.group(1))
        except Exception: return None
    match = re.search(r'(\{.*?\})', texto, re.DOTALL)                    # 2)
     fallback: llaves
    if match:
        try:    return json.loads(match.group(1))
        except Exception: return None
    return None
```

Listado 5.12: reset() reinicia el estado pero preserva la caché de veredictos de sesión.

```
def reset(self):
    self.turns = 0
    self.controles_a_evaluar = []
    self.indice_control_actual = 0
    self.indice_requisito_actual = 0
    self.requisitos_control_actual = []
    self.respuestas_requisitos = {}
    self.evaluaciones_controles = {}
    # self.cache_veredictos NO se limpia: persiste durante toda la sesión.
```

Listado 5.13: Extracto real (anonimizado) de una traza de auditoría.

```
{
  "session_id": "",
  "evaluaciones": [{
    "control_legislativo": { "id": "NIS2 Art. 21.2 e)", "titulo": "Seguridad en la
     adquisición" },
    "resultado_st": {
      "cumple": false, "puntuacion": 2,
      "razon": "Op.mon.1: la empresa no aportó evidencia documental de la
     configuración de IPS/IDS"
    },
    "controles_tecnicos": [
      { "control_id": "Mp.com.2",   "cumple": true,  "similitud": 0.785,
        "razon_llm": "Aporta evidencia de uso de VPN y de los algoritmos cifrados
     autorizados" },
      { "control_id": "Op.mon.1",   "cumple": false, "similitud": 0.749,
        "razon_llm": "Sin evidencia documental de la configuración y uso de IPS/IDS"
      }
    ]
  }]
}
```

Traza de auditoría

Al cerrar cada control legislativo, `_guardar_audit_trail` vuelca un **único fichero JSON por sesión** (en `ST/audit_trail/`) que se amplía con cada control. El Listado 5.13 muestra un extracto real (anonimizado) de la estructura: el resultado que el módulo del ST aporta para el control legislativo y el detalle de los controles técnicos auditados, cada uno con su veredicto, la razón del modelo y la similitud que motivó su selección. El veredicto que aquí se persiste es el *técnico*; el global lo emite el SN (Sección 4.9.4).

El juez como salvaguarda

El juez anti-falsos-positivos (Sección 4.8.2) está aislado en `juez_controles.py`. Su *prompt* de sistema le fija criterios explícitos de inclusión y exclusión y le obliga a **razonar antes de decidir** (qué medida exige el artículo, qué evidencia la demostraría y qué candidato la evalúa) para responder después únicamente con JSON:

En código, lo relevante son sus **salvaguardas**: si hay dos candidatos o menos no se invoca, se aplica un tope de seguridad y, ante cualquier fallo, degrada con elegancia a los tres candidatos de mayor similitud.

La Figura 5.3 muestra el resultado de todo lo anterior en la interfaz: una pregunta generada por el agente a partir de un requisito técnico, con su párrafo de pista final.

Listado 5.14: *Prompt* del juez: criterios y razonamiento previo a la selección (extracto).

```
CRITERIO DE INCLUSIÓN - un control es pertinente si:
- Su título describe exactamente la actividad o medida que exige el artículo.
- Auditarlo responde a "¿cumple la empresa con este artículo?".
- Sin él, el artículo quedaría sin evidencia técnica suficiente.

CRITERIO DE EXCLUSIÓN - descártalo si:
- Solo comparte vocabulario genérico de seguridad pero evalúa otra práctica.
- Cubre un dominio adyacente, no el que el artículo requiere específicamente.

ANTES DE SELECCIONAR, razona brevemente:
  1. ¿Qué medida técnica concreta exige este artículo?
  2. ¿Qué evidencia habría que presentar para demostrarlo?
  3. ¿Cuál de los candidatos evalúa ESA evidencia concreta?

Responde SOLO con un JSON válido:
{ "seleccionados": ["ID1", "ID2", ...], "razon": "por qué estos y no otros" }
```

Listado 5.15: Salvaguardas del juez: omisión, tope y degradación elegante.

```
if len(candidatos) <= 2:                        # no hace falta filtrar
    return candidatos
...
match = re.search(r'\{.*?\}', contenido, re.DOTALL)
if not match:                                   # respuesta no parseable
    return candidatos[:3]                        # -> degradación: top-3
...
if len(ids_seleccionados) > TOPE_SEGURIDAD:    # JUEZ_TOPE_MAXIMO = 10
    ids_seleccionados = ids_seleccionados[:TOPE_SEGURIDAD]
```

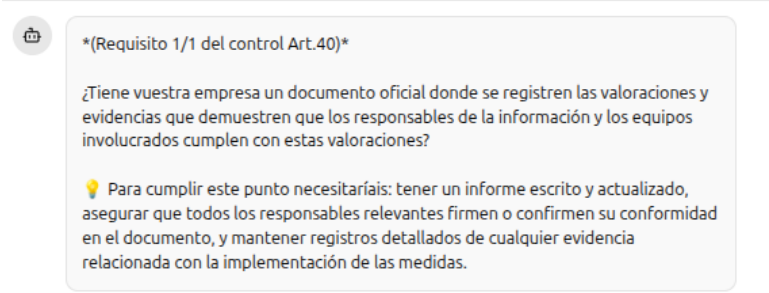

Figura 5.3: Pregunta real generada por el agente conversacional a partir de un requisito técnico, traducida a lenguaje de negocio y cerrada con el párrafo "Para cumplir este punto necesitaríais: ...". Materializa el *prompt* del Listado 5.9.

Listado 5.16: Endpoints del servicio ST.

```
@app.get("/health")
def health():
    return {"status": "ok", "service": "ST", "ollama": ST_OLLAMA_HOST,
            "agente_real": _agente_real is not None}

@app.post("/chat", response_model=ChatResponse)
def chat(req: ChatRequest):
    return _real_chat(req)
```

Listado 5.17: Modelo de perfil con alias *frontend*/*backend*.

```
class CompanyProfile(BaseModel):
    model_config = ConfigDict(populate_by_name=True)
    applies_dora: bool = Field(False, alias="appliesDORA")
    applies_nis2: bool = Field(False, alias="appliesNIS2")
    nis2_type:    str  = Field("None", alias="nis2Type")
    applies_ens:  bool = Field(False, alias="appliesENS")
    sector: str = ""; size: str = ""
    categoria_global: str = "BÁSICA"
    dimensiones: dict = {"C": "BÁSICA", "I": "BÁSICA", "T": "BÁSICA", "A": "BÁSICA",
      "D": "BÁSICA"}
```

5.4. Servicio ST (FastAPI)

El agente se expone como un servicio FastAPI (`st_service.py`) con una superficie deliberadamente **mínima**: solo dos *endpoints*, `GET /health` y `POST /chat`.

Modelos y *alias*. Los modelos Pydantic validan el contrato. `CompanyProfile` resuelve la divergencia de nomenclatura entre el *frontend* (inglés *camelCase*) y la lógica interna mediante **alias**, habilitando además el poblado por nombre para aceptar ambas formas (Sección 4.12):

Delimitación de controles

Como el módulo del ST es conversacional pero el protocolo HTTP es sin estado, el servicio necesita saber cuándo empieza un control nuevo. La convención es simple: un **historial vacío** señala el inicio de un control, momento en que se reinicia el agente y se fija el identificador de sesión para la traza de auditoría.

Listado 5.18: Historial vacío = nuevo control: reset del agente.

```
if not req.history or len(req.history) == 0:
    _agente_real.reset()
    _agente_real._session_id_override = req.session_id   # para el audit trail
```

Listado 5.19: Avance con salto automático de artículos sin evaluación técnica (extracto).

```
async def _setup_siguiente_control_evaluable(session_id: str, session: dict):
    while session["current"] < len(session["controls"]):
        control = session["controls"][session["current"]]
        st_setup = await _post(f"{ST_URL}/chat", {          # historial vacío = reset
            "session_id": session_id, "control": control,
            "message": "", "history": [], "profile": session["profile"]})

        if st_setup.get("sin_control_tecnico"):
            session["results"].append({ "control": control, "sin_control_tecnico": True })
            session["current"] += 1                         # salto sin interrogar al usuario
            continue
        return control, st_setup
    return None, None
```

5.5. Capa de conexión e instrumentación

El módulo del ST no se invoca directamente: lo orquesta el **Coordinador** (`coordinator.py`), también FastAPI, que mantiene el estado de sesión en memoria y conduce una conversación con el módulo del ST **por cada control legislativo** de la lista que devuelve el SN (Sección 4.12).

Bucle de orquestación y salto de controles

Cuando el módulo del ST marca un **control legislativo** como terminado, el Coordinador guarda su resultado y avanza al siguiente. Para solicitar la primera pregunta del nuevo artículo, envía un **historial vacío**, reutilizando así la convención del Listado 5.18. Si un artículo normativo resulta no tener controles técnicos evaluables (el ST devuelve `sin_control_tecnico = True`), el Coordinador lo anota en los resultados y avanza en bucle al siguiente sin requerir interacción del usuario, agilizando la evaluación. Agotados todos los controles legislativos, remite el conjunto de resultados al SN para el veredicto final.

Instrumentación de cableado (WIRE_DEBUG)

Para depurar la integración entre servicios sin saturar los *logs* en producción, el Coordinador incorpora una instrumentación opcional gobernada por la variable de entorno `WIRE_DEBUG`. Cuando está activa, registra cada carga útil que sale y entra (`[WIRE OUT]` / `[WIRE IN]`), serializada y truncada a un máximo configurable de caracteres para no volcar *embeddings* completos:

Listado 5.20: Instrumentación opcional del tráfico entre servicios.

```
WIRE_DEBUG = os.getenv("WIRE_DEBUG", "false").lower() == "true"

async def _post(url, data):
    if WIRE_DEBUG: log.info("[WIRE OUT -> %s]\n%s", _wire_target(url), _wire_preview
     (data))
    async with httpx.AsyncClient(timeout=TIMEOUT) as client:
        resp = await client.post(url, json=data); resp.raise_for_status()
        response = resp.json()
    if WIRE_DEBUG: log.info("[WIRE IN <- %s]\n%s", _wire_target(url), _wire_preview(
     response))
    return response
```

Listado 5.21: Arranque secuencial de los tres servicios (extracto).

```
python3 sn_service.py     > logs/sn.log          2>&1 &  SN_PID=$!
python3 st_service.py     > logs/st.log          2>&1 &  ST_PID=$!
python3 coordinator.py    > logs/coordinator.log 2>&1 &  COORD_PID=$!
echo "$SN_PID $ST_PID $COORD_PID" > .service_pids
```

Tiempos de espera y degradación

Dado que la Fase 1 usa el modelo grande (más lento), todas las llamadas `httpx` fijan un **tiempo de espera amplio** (`TIMEOUT = 200` segundos). Los fallos se traducen en códigos HTTP explícitos (502/503).

Arranque y configuración

El conjunto se levanta con `start_services.sh`, que activa (o crea) el *virtualenv*, libera los puertos, arranca en orden SN, ST y Coordinador redirigiendo cada salida a `logs/{sn,st,coordinator}.log` y guarda los PID para poder pararlos:

Toda la configuración sensible (URIs y credenciales de Neo4j, *host* y modelos de Ollama, puertos) se externaliza en un fichero `.env` cargado con `python-dotenv`, nunca versionado. La Tabla 5.3 resume las variables principales con los valores reservados redactados.

Como comprobación de que el conjunto está operativo, el arranque deja los tres servicios escuchando y el Coordinador expone un `/health` que verifica, a su vez, la disponibilidad del módulo del ST y del SN (Listado 5.22); esta comprobación es el punto de partida de cualquier sesión de evaluación.

Cuadro 5.3: Variables de entorno principales (valores sensibles redactados).

Variable	Valor / descripción
COORDINATOR_PORT / ST_SERVICE_PORT / SN_SERVICE_PORT	5000 / 5001 / 5002
ST_OLLAMA_HOST	http://localhost:11435 (boca del túnel SSH a la UGR)
ST_OLLAMA_MODEL	qwen2.5:14b (agente)
JUEZ_OLLAMA_MODEL	qwen2.5:32b (juez)
NEO4J_URI_ST	neo4j+s://<instancia>.databases.neo4j.io
NEO4J_USER_ST / NEO4J_PASSWORD_ST	<oculto>
WIRE_DEBUG	true|false — instrumentación del tráfico entre servicios
LOG_LEVEL	INFO

Listado 5.22: Arranque de los servicios y comprobación de salud del conjunto.

```
$ bash start_services.sh
  [1/3] SN Service  -> http://localhost:5002  (log: logs/sn.log)
  [2/3] ST Service  -> http://localhost:5001  (log: logs/st.log)
  [3/3] Coordinator -> http://localhost:5000  (log: logs/coordinator.log)
  Servicios arrancados (PIDs: SN, ST, Coord)

$ curl http://localhost:5000/health
{"status": "ok", "services": {"SN": "ok", "ST": "ok"}}
```

Capítulo 6

Pruebas y validación

Este capítulo verifica que el módulo del Sistema Técnico hace lo que debe: que, dado el perfil de una empresa y un control, infiere el conjunto de requisitos que las tres normativas realmente exigen. El objeto central de la validación es el **motor de inferencia** (el Principio de Máximos resuelto sobre el grafo, Capítulo 4), por dos razones: es la pieza que concentra toda la lógica normativa y es **determinista**, lo que permite comprobarla con precisión. Las etapas apoyadas en modelos de lenguaje (el juez de la Fase 1 y el juicio de cumplimiento de cada control) no son deterministas y quedan, por su naturaleza, fuera de una validación por igualdad exacta; de ellas se verifica únicamente que sus **salvaguardas** contienen cualquier respuesta anómala del modelo.

La estrategia de validación sigue un enfoque progresivo. Se parte de pruebas unitarias que aíslan las funciones deterministas del motor, para pasar luego a una validación funcional exhaustiva del Principio de Máximos frente a las normativas reales (las reglas del PCE-NIS2, el *crosswalk* de DORA y la guía CCN-STIC-892). De esta forma se comprueba que el motor satisface la norma, no que simplemente reproduce su propio comportamiento. El proceso concluye con pruebas de integración que aseguran el funcionamiento robusto del flujo completo orquestado por el Coordinador.

6.1. Pruebas unitarias del motor

Antes de validar el comportamiento normativo del motor en su conjunto (Sección 6.2), un primer nivel de pruebas verifica de forma **aislada** sus piezas deterministas. Las seis pruebas residen en el directorio `ST/tests/` y cada una invoca directamente la pieza que comprueba con entradas controladas, contrastando su salida con el valor esperado mediante aserciones: la puntuación (`test_puntuacion.py`, U1), la extracción de JSON (`test_extraer_`

Cuadro 6.1: Pruebas unitarias de las piezas deterministas del motor.

Prueba	Propiedad verificada	Comp.
U1	Puntuación determinista 1–10: todas las ramas de la Tabla 4.6 (los tres tramos con fallo esencial, los siete umbrales sin él y el caso sin controles), incluida la comprobación de que un requisito esencial *cumplido* no penaliza.	12
U2	Extracción robusta de JSON (Listado 5.11): bloque Markdown, objeto crudo entre texto libre, y degradación a `None` ante texto sin JSON o JSON malformado.	5
U3	*Caché* de veredictos de sesión: el veredicto se almacena con su origen, se reutiliza sin volver a consultar el grafo ni el modelo, y `reset()` lo conserva a lo largo de varios controles legislativos.	5
U4	Deduplicación base/refuerzo en la preparación: si el juez devuelve un refuerzo (`Op.pl.2.r2`), se evalúa su control base, y el control base duplicado se omite por estar ya cubierto.	2
U5	Salvaguardas del juez (Listado 5.15), con un modelo falso: con dos candidatos o menos el modelo no llega a invocarse; una respuesta no parseable, una excepción del modelo o una selección sin identificadores válidos degradan al top-3 por similitud; y una selección excesiva se recorta al tope de seguridad.	6
U6	Formato de las observaciones del veredicto: la razón agregada enumera los controles fallidos con su motivo, en los tres escenarios (fallo esencial, fallo mixto y fallo solo de no esenciales).	3
	Total	**33**

`json.py`, U2), la *caché* de veredictos (`test_cache_veredictos.py`, U3), la deduplicación base/refuerzo (`test_fase2_rapido.py`, U4), las salvaguardas del juez (`test_juez_salvaguardas.py`, U5) y el formato de las observaciones (`test_observaciones.py`, U6).

La estrategia distingue dos planos según las dependencias que necesita cada pieza. Las funciones puras y las defensas frente al modelo de lenguaje se prueban **sin grafo y sin modelo**, sustituyendo esas dependencias externas por **dobles de prueba** (objetos simulados que devuelven respuestas fijadas por el test), de modo que la *suite* se ejecuta en segundos y puede correr en cualquier entorno. La única excepción es la deduplicación base/refuerzo (U4), que depende de la consulta de inferencia y se ejerce contra la instancia Neo4j real. La Tabla 6.1 resume las seis pruebas; en conjunto suman **33 comprobaciones, todas satisfechas**.

Las pruebas U2 y U5 cubren la **frontera con el modelo de lenguaje**, el único punto del motor donde puede entrar una respuesta arbitraria. Verificar que toda salida malformada degrada de forma controlada (a `None` con veredicto conservador en la evaluación, al top-3 por similitud en el juez) es lo que garantiza que un fallo del modelo nunca tumba la evaluación, tal como se diseñó en la Sección 4.5.

6.2. Validación funcional del Principio de Máximos

El núcleo del módulo del Sistema Técnico, el motor de inferencia que resuelve el Principio de Máximos sobre el grafo (Sección 4.8), es **determinista**: dados un perfil de empresa y un control, la combinación de suelos, refuerzos y exclusiones de las tres normativas produce siempre el mismo conjunto de requisitos. Esa propiedad lo hace idóneo para una validación sistemática mediante **baterías de pruebas basadas en invariantes**.

Comprobar que el motor acierta exige contrastar lo que produce con un **resultado de referencia**, y la decisión metodológica más relevante es de dónde se obtiene ese resultado. No se toma de la **salida del propio motor** (eso solo verificaría que el motor se comporta igual que sí mismo, y daría por buenos sus propios errores), sino de las **especificaciones normativas de origen**: el fichero de reglas del PCE-NIS2 (`reglas_pce_nis2.json`), el *crosswalk* ENS–DORA (`dora_mapeo_ens.json`) y la tabla de aplicabilidad del documento CCN-STIC-892. Así, las pruebas comprueban que el motor **satisface la norma**, no que reproduce su propio comportamiento, y son independientes de la implementación *Cypher* que validan.

Visto de extremo a extremo, la validación cierra el recorrido **de la ley a la salida del sistema** en dos tramos. El primero confirma que los **datos del grafo reflejan fielmente la norma**: las reglas codificadas se contrastan contra el documento oficial, como en la verificación de los 27 suelos dimensionales frente a la tabla del CCN-STIC-892 (Sección 6.2.3). El segundo, automático y reproducible, confirma que el **motor, partiendo de esos datos, produce la salida que las reglas imponen**, y es el cometido de las baterías que se describen a continuación. El reparto en dos tramos es necesario porque el documento normativo no es legible por una máquina: el eslabón ley–datos se verifica de forma manual una sola vez, y la comprobación datos–salida es la que automatizan las baterías. Cada una recorre los controles con regla especial, genera los perfiles relevantes, ejecuta la consulta de inferencia real contra el grafo y contrasta la salida con la invariante derivada de la especificación.

6.2.1. Batería del PCE-NIS2

La batería de la NIS2 (`test_pce_nis2_892.py`) traduce a invariantes verificables las cuatro familias de reglas que el PCE-NIS2 [3] impone sobre las medidas del ENS: exclusiones, suelos, refuerzos forzados y exclusiones de refuerzo diferenciadas por tipo de entidad. La Tabla 6.2 resume las seis invariantes y su cobertura; la *suite* comprende **142 comprobaciones, todas satisfechas**.

Cuadro 6.2: Invariantes verificadas en la batería funcional del PCE-NIS2.

Inv.	Propiedad verificada (derivada del 892)	Comp.
T1	`excluido_pce_nis2`: la medida excluida no recibe exigencia de la NIS2 para ninguna entidad (`Mp.info.2`).	2
T2	`excluido_esenciales`: una entidad *esencial* no recibe exigencia NIS2 sobre la medida excluida.	2
T3	`excluido_importantes`: una entidad *importante* no recibe exigencia NIS2 sobre la medida excluida.	4
T4	`importante_excl`: el refuerzo excluido falta *solo* para entidades importantes, y permanece para esenciales y para sistemas ENS puros.	21
T5	`esencial_/importante_requiere_refuerzos`: el refuerzo forzado por la NIS2 aparece aun cuando la categoría ENS sería insuficiente.	40
T6	Monotonía: el conjunto de requisitos de una entidad esencial contiene al de la misma organización sin NIS2 (un suelo solo añade).	73
	Total	**142**

La invariante **T4** es la más exigente porque captura la interacción entre los refuerzos del ENS y las exclusiones de refuerzo de la NIS2. El caso canónico es el control `Op.exp.1` (inventario de activos), sobre el que el 892 fuerza los refuerzos `r1`–`r4` a las entidades esenciales pero exige únicamente `r4` (excluyendo `r1`, `r2` y `r3`) a las importantes: la prueba confirma que el motor entrega exactamente esos conjuntos según el tipo de entidad. La invariante **T6** se comprueba sobre los 73 controles, lo que la convierte en una red de seguridad global frente a regresiones.

6.2.2. Batería de DORA

DORA no introduce suelos graduados por categoría, sino **refuerzos innegociables** que se activan cuando la entidad queda sujeta al Reglamento (Sección 3.2.1). La batería correspondiente (`test_dora.py`) deriva sus expectativas del *crosswalk* `dora_mapeo_ens.json` y verifica esa mecánica de transversalidad. Dado que la precedencia *lex specialis* se resuelve en la fase de clasificación, de modo que una entidad sujeta a DORA llega al motor sin exigencias NIS2 concurrentes (Sección 3.3), las pruebas se ejecutan sobre perfiles con DORA activo y sin tipo NIS2. La Tabla 6.3 recoge las cuatro invariantes; la *suite* comprende **135 comprobaciones, todas satisfechas**.

6.2.3. Batería de la ponderación dimensional

La tercera batería valida la **ponderación por dimensión del riesgo propio** (Sección 4.8.3), y lo hace en tres planos complementarios: la fidelidad de los datos a la fuente normativa, las invariantes del motor real y una exploración exhaustiva de perfiles.

Cuadro 6.3: Invariantes verificadas en la batería funcional de DORA.

Inv.	Propiedad verificada (derivada del *crosswalk*)	Comp.
D1	Innegociabilidad: cada refuerzo que DORA exige aparece aun con categoría ENS BÁSICA.	47
D2	Sin DORA, un refuerzo de activación exclusivamente sectorial desaparece de la evaluación.	14
D3	Independencia de categoría: el refuerzo DORA está presente por igual en BÁSICA y en ALTA (no se gradúa).	47
D4	Trazabilidad: los requisitos del control activados por DORA llevan su origen normativo.	27
	Total	**135**

Fidelidad a la fuente primaria

Los 27 controles cuyo suelo NIS2 se expresa de forma dimensional (`minimo_nivel`) se contrastaron uno a uno contra la tabla de aplicabilidad del documento CCN-STIC-892, comparando simultáneamente el **conjunto de dimensiones afectadas**, el **nivel exigido** (Bajo/Medio/Alto) y la **aplicabilidad por tipo de entidad**: los **27 coinciden en los tres ejes**. La comprobación exigió fijar con cuidado la semántica de la tabla original, cuya cabecera ordena las columnas como *Importantes* | *Esenciales*, el orden inverso al habitual, lo que resulta determinante en los controles asimétricos: `mp.info.3` y `mp.info.4` obligan únicamente a las entidades esenciales, mientras que `mp.info.5` obliga únicamente a las importantes, y el grafo refleja exactamente esa asimetría. La verificación confirma además que el 892 reserva la expresión dimensional para las medidas que el ENS marca como dimensión-específicas: las cuatro medidas de continuidad sobre la disponibilidad, el cifrado de comunicaciones sobre la confidencialidad o la trazabilidad de `mp.info.4` sobre la dimensión de trazabilidad, entre otras. Esto respalda la decisión de diseño de la Sección 4.8.3.

Invariantes sobre el motor real

La batería `test_evaluacion_por_dimension.py` ejecuta la consulta de inferencia real contra el grafo y verifica las cinco invariantes de la Tabla 6.4; la *suite* comprende **36 comprobaciones, todas satisfechas**.

La invariante **G1** es la **regla de oro** y garantiza la *compatibilidad* del cambio: un control que afecta a las cinco dimensiones se evalúa exactamente igual en los dos modos. La razón es que el ENS define la categoría global de un sistema como el **máximo de los niveles de sus cinco dimensiones**; por tanto, para un control que las abarca todas, tomar ese máximo (modo dimensional) o usar directamente la categoría global (modo anterior) da forzosamente el mismo valor. Verificarla asegura que la ponderación por

Cuadro 6.4: Invariantes verificadas en la batería de la ponderación dimensional.

Inv.	Propiedad verificada	Comp.
G1	Regla de oro: un control que afecta a las cinco dimensiones produce exactamente el mismo resultado con el perfil dimensional que con la categoría global equivalente.	6
G2	Monotonía: reducir el nivel de una dimensión nunca añade requisitos; el conjunto evaluado con un perfil dimensional es subconjunto del evaluado con la categoría global.	16
G3	Efecto real: existen controles de dimensiones parciales cuyo conjunto de requisitos se ajusta estrictamente al nivel de su dimensión, de modo que la ponderación por dimensión **cambia el resultado de verdad** y no es una distinción sin efecto.	6
G4	Los suelos de la NIS2 no se ven afectados: un control con suelo dimensional NIS2 recibe la misma exigencia aunque el nivel propio de la empresa en esa dimensión sea bajo.	2
G5	Reducción al caso global: un perfil sin niveles dimensionales equivale a uno con la categoría global en las cinco dimensiones.	6
	Total	**36**

dimensión solo **afina los controles parciales** (los que afectan a menos de cinco dimensiones) y deja intactos los que dependen de todas: si alguno de estos últimos cambiara entre un modo y otro, sería un defecto. La invariante **G4** captura la propiedad estructural más importante del diseño: como el peso de la empresa es un término independiente dentro del máximo, cambiar su forma de cálculo *no puede* rebajar un suelo normativo, pues la NIS2 y DORA siguen entrando en la comparación con sus propios pesos. Un ejemplo ilustrativo de **G2/G3**: el control `Op.cont.1` (análisis de impacto), que afecta únicamente a la disponibilidad, mantiene sus seis requisitos cuando la empresa tiene la dimensión D en nivel Alto pero queda sin exigencias cuando su riesgo alto reside en otra dimensión; a la inversa, `Mp.com.2` (protección de la confidencialidad) se audita por el nivel de C con independencia de la D.

Exploración exhaustiva de perfiles

Como complemento, una prueba sistemática (`test_evaluacion_por_dimension_neo4j.py`) ejerce el **motor real sobre el grafo Neo4j**, invocando la misma función de inferencia que emplea producción, sin réplica ni simulación. La matriz de prueba recorre **12 perfiles** que varían la dimensión que concentra el riesgo, la categoría resultante, el tipo de entidad NIS2 y la sujeción al ENS; cada perfil se evalúa en sus dos modos (ponderación por dimensión y por categoría global equivalente) a lo largo de los **222 controles del catálogo**, lo que suma **5 328 evaluaciones** contra la base de

Cuadro 6.5: Exploración exhaustiva sobre el grafo real (`test_evaluacion_por_dimension_neo4j.py`). Cada fila es un perfil evaluado en sus dos modos a lo largo de los 222 controles. *Regla de oro*: los 131 controles que afectan a las cinco dimensiones coinciden en ambos modos (✓ = 0 discrepancias). *Cambian*: número de controles cuyo conjunto de requisitos difiere entre la ponderación por dimensión y la de categoría global.

Perfil (dónde se concentra el riesgo)	**NIS2**	**ENS**	**Regla de oro**	**Cambian**
ALTA, pico en C	Esencial	sí	✓	16
ALTA, pico en D	Esencial	sí	✓	26
ALTA, pico en I	Esencial	sí	✓	13
ALTA, solo T alto	Esencial	sí	✓	30
ALTA, pico en A	Importante	sí	✓	24
MEDIA, picos en C y D	Esencial	sí	✓	11
MEDIA, solo D	Importante	sí	✓	20
BÁSICA, todo al mismo nivel	Esencial	sí	✓	0
ALTA, todo al mismo nivel	Esencial	sí	✓	0
ALTA, pico en C	Ninguno	sí	✓	25
ALTA, pico en C	Esencial	no	✓	0
ALTA, pico en T	Importante	sí	✓	14

conocimiento.

La regla de oro se verifica de forma directa: los **131 controles que afectan a las cinco dimensiones** producen un resultado idéntico en ambos modos para los doce perfiles, **1 572 comprobaciones sin una sola discrepancia**. Los casos extremos se comportan como cabe esperar: con todas las dimensiones al mismo nivel no hay diferencia alguna con la ponderación por categoría, y sin sujeción al ENS el lado de la empresa no interviene. El efecto de la ponderación dimensional es, además, real y apreciable: según en qué dimensión resida el pico de riesgo, entre 11 y 30 controles ajustan su conjunto de requisitos respecto a la evaluación por categoría global. La Tabla 6.5 recoge los doce perfiles con su resultado.

6.2.4. Defectos detectados y corregidos

El valor de una batería de validación se mide por su capacidad para descubrir discrepancias reales; las baterías de la NIS2 y de DORA cumplieron ese cometido durante su desarrollo:

- **Alcance de las exclusiones de refuerzo (NIS2).** La invariante T4 reveló que la exclusión de refuerzos definida para entidades importantes (`importante_excl`) se estaba aplicando también a entidades esenciales y a sistemas ENS puros, suprimiendo refuerzos legítimos del ENS. La consulta de inferencia se corrigió para condicionar esa

exclusión al tipo de entidad, y la invariante quedó satisfecha en los cinco controles afectados.

- **Consistencia de identificadores (DORA).** Las invariantes D1 y D3 detectaron dos referencias del *crosswalk* ENS–DORA (`Org.3` y `Op.mon.3`) que apuntaban a refuerzos cuyo identificador no coincidía con el del catálogo del ENS, por lo que nunca llegaban a activarse. Corregido el mapeo, ambos refuerzos se incorporan a la evaluación.

Que estas pruebas se deriven de la especificación normativa, y no del motor, es lo que permitió que afloraran defectos que un oráculo construido a partir del propio sistema habría dado por correctos.

6.3. Pruebas de integración con el Coordinador

Por encima del motor, la integración entre el módulo del ST y el Coordinador se apoya en un contrato deliberadamente estrecho (Sección 4.12): el Coordinador abre una conversación con el módulo del ST **por cada control legislativo**, delimita el inicio de cada uno enviando un **historial vacío** (la convención que reinicia el agente, Listado 5.18) y, cuando el módulo del ST marca el control como terminado, guarda su veredicto y avanza al siguiente hasta agotar la lista, momento en que remite el conjunto de resultados al SN. Dos mecanismos protegen ese flujo en la propia frontera: la validación de los contratos por los modelos `Pydantic` con sus *alias* (Sección 4.12), que rechaza una carga mal formada antes de que llegue al motor, y la **degradación** del Coordinador, que ante un fallo del SN devuelve los resultados técnicos del módulo del ST en lugar de abortar la sesión.

La verificación de extremo a extremo de este flujo (una evaluación completa con los cuatro servicios en marcha, desde el perfil de la empresa hasta el informe final) requiere el entorno desplegado al completo (*frontend*, Coordinador, SN y ST) y se documenta, con la traza real de una ejecución, en el Anexo B.

Para ejercitar el flujo completo de forma reproducible sin teclear manualmente cada respuesta, se desarrolló un **usuario simulado**: un modelo Ollama ligero (`qwen2.5:7b`) que hace de empresa y contesta las preguntas del agente de evaluación, consumiendo la misma API que el *frontend*. El simulador no modifica ni reimplementa la lógica del Coordinador ni del ST; su único propósito es automatizar la parte del usuario para poder observar el recorrido completo (un control legislativo tras otro, hasta el veredicto) sin intervención manual. Se ofrecen varios perfiles de respuesta (*personas*): una PYME con lo básico cubierto, otra con madurez a medias y otra sin conocimientos, además de un perfil experto que cumple todo, para provocar

veredictos tanto de cumplimiento como de incumplimiento y observar ambos caminos.

Se trata de instrumentación de validación, no de una funcionalidad del producto. Su uso se ilustra en el Anexo B, donde se emplea para conducir una evaluación completa y generar la traza de auditoría de extremo a extremo.

Capítulo 7

Conclusiones y trabajo futuro

Como cierre del proyecto, este capítulo realiza un balance final del trabajo desarrollado. Se evalúa el grado de cumplimiento de los objetivos iniciales y se contextualiza la herramienta frente al actual marco regulatorio de la inteligencia artificial. Además, se analizan con espíritu crítico las limitaciones técnicas encontradas durante la implementación, proponiendo líneas de trabajo futuro para solventarlas y reflexionando sobre las posibilidades de explotación comercial del sistema en un entorno real.

7.1. Consecución de los objetivos

Esta sección revisa en qué medida se han alcanzado los objetivos específicos planteados en la Sección 2.1. El balance es positivo: los nueve objetivos se han **alcanzado por completo** y se materializan en los capítulos de Diseño, Implementación y Evaluación, como resume la Tabla 7.1.

Cuadro 7.1: Grado de consecución de los objetivos específicos.

OE	**Objetivo específico**	**Estado**	**Cap./§**
OE1	Grafo de conocimiento normativo en Neo4j	Logrado	4.7
OE2	Inferencia de controles por búsqueda semántica	Logrado	4.2
OE3	Filtrado de falsos positivos (juez LLM)	Logrado	4.8.2
OE4	Principio de Máximos resuelto en *Cypher*	Logrado	4.8.3
OE5	Auditoría conversacional en lenguaje de negocio	Logrado	5.3
OE6	Veredicto trazable y puntuación (1–10)	Logrado	5.3
OE7	Clasificación CITAD de la categoría ENS	Logrado	4.11
OE8	Integración con el resto del sistema vía API	Logrado	5.5
OE9	Validación sistemática frente a invariantes normativas	Logrado	6.2

Objetivo general

En conjunto, el objetivo general (Sección 2.1.1), diseñar, implementar e integrar el módulo del Sistema Técnico, apoyado en un grafo de conocimiento, capaz de inferir los requisitos aplicables, evaluarlos y emitir un veredicto trazable, queda **cumplido** en su diseño, su implementación, su integración y su validación (OE1–OE9). Las limitaciones de la solución y las líneas que la continúan se desarrollan en las Secciones 7.3 y 7.4.

7.2. Consideraciones legales y éticas

El Capítulo 3 caracterizó las normas que el módulo del Sistema Técnico *evalúa* (ENS, NIS2 y DORA). Procede ahora un eje distinto: la norma que regula *al propio ST* en tanto que sistema de inteligencia artificial. El **Reglamento (UE) 2024/1689** (Reglamento de Inteligencia Artificial, RIA), en vigor desde 2024, ordena los sistemas de IA según un **enfoque basado en el riesgo** con cuatro niveles (prácticas prohibidas, alto riesgo, sistemas con obligaciones de transparencia y riesgo mínimo) y modula sus exigencias en consecuencia [18, 12].

Clasificación del módulo del Sistema Técnico

El módulo del ST es un **asistente conversacional** que dialoga con un responsable de negocio; como tal, se sitúa en el nivel de **sistemas con obligaciones de transparencia**, donde la pirámide de riesgos ubica expresamente a los *chatbots* [12]. No pertenece a las categorías superiores: no incurre en ninguna de las prácticas prohibidas (manipulación cognitiva, puntuación social, vigilancia masiva), ni constituye un sistema de **alto riesgo**, reservado a usos como la selección de personal, la justicia, la admisión educativa o las infraestructuras críticas, pues el módulo del ST asesora sobre cumplimiento sin adoptar decisiones automatizadas que afecten a los derechos fundamentales de las personas. La única exigencia directa que de ello se deriva es **informar al usuario de que interactúa con un sistema de IA**, la obligación de transparencia que el **artículo 50** del Reglamento impone a los sistemas destinados a interactuar con personas físicas. El etiquetado de contenidos sintéticos (*ultrafalsificaciones*) no le aplica, ya que el módulo del ST no genera medios artificiales.

Alineación del diseño con el espíritu del Reglamento

Aunque solo le obliga la transparencia, el diseño del módulo del ST anticipa principios de **rendición de cuentas** que el RIA promueve para

los sistemas de mayor riesgo: el veredicto del motor de inferencia es **determinista** (el Principio de Máximos), cada requisito conserva su **origen normativo** y cada evaluación deja una **traza de auditoría**, de modo que toda decisión es reconstruible y explicable; y la valoración mantiene al **humano en el bucle**, pues nace de las respuestas del usuario y desemboca en un informe que un responsable interpreta. Determinismo, trazabilidad y supervisión humana son las garantías sobre las que pivota el Reglamento.

Plano nacional

En España, el **Proyecto de Ley Orgánica para el buen uso y la gobernanza de la inteligencia artificial**, aprobado por el Consejo de Ministros el 26 de mayo de 2026, adapta el RIA al ordenamiento interno: designa las autoridades de vigilancia, con la AESIA como árbitro central apoyado en las sectoriales ya existentes (Banco de España, CNMV, AEPD, entre otras), y fija el régimen sancionador, sin alterar la clasificación del módulo del ST [12]. Se trata, en todo caso, de un **proyecto de ley en tramitación**, y no de derecho vigente, en el momento de redactar esta memoria.

7.3. Dificultades encontradas

Evaluación basada en declaraciones

La evaluación se construye sobre las respuestas que el usuario proporciona en lenguaje natural durante la conversación. En su forma actual, el sistema es un asistente puramente conversacional: no ingiere ni comprueba las evidencias (documentos, configuraciones, registros) que sustentarían cada respuesta, sino que toma como válida la información declarada. En consecuencia, el veredicto refleja el cumplimiento *declarado* y no necesariamente el *implementado*: una respuesta errónea o sobreestimada por parte del usuario se traslada al resultado. Esta decisión es coherente con el destinatario, un responsable de negocio y no un auditor técnico, pero acota la garantía que el sistema puede ofrecer. La superación de esta limitación se aborda en la Sección 7.4.

Naturaleza no determinista de los componentes basados en modelos de lenguaje

Mientras que el motor de inferencia (el Principio de Máximos) y la puntuación final son funciones deterministas, las dos etapas apoyadas en modelos de lenguaje (la selección de controles del juez y el juicio de cumplimiento

de cada requisito) no lo son en sentido estricto. Se ejecutan con temperatura cero para maximizar la reproducibilidad, lo que reduce la variabilidad pero no la elimina: ante una misma entrada, el resultado puede no ser idéntico entre ejecuciones.

Ausencia de persistencia

El estado de la sesión reside en memoria del Coordinador y no se persiste en ningún almacén. Las evaluaciones se pierden al reiniciar el servicio, lo que impide consultar el histórico de una empresa, retomar una sesión interrumpida o explotar los resultados a posteriori más allá de la traza de auditoría que se vuelca por sesión. Su solución se recoge también en la Sección 7.4.

Procedencia limitada de los mapeos entre marcos

La proyección de NIS2 y DORA sobre los controles del ENS, que sustenta el Principio de Máximos, descansa en correspondencias cuya disponibilidad y oficialidad son desiguales. El mapeo NIS2↔ENS se apoya en el Perfil de Cumplimiento Específico PCE-NIS2 (CCN-STIC-892) en su versión de 2024 (Sección 3.2.2), que fue retirada del portal del CCN-CERT durante la elaboración de este trabajo, a la espera de una nueva versión alineada con el Reglamento de Ejecución (UE) 2024/2690 y con la transposición de la Directiva. El mapeo ENS↔DORA, por su parte, no procede de un *crosswalk* normativo oficial, inexistente a día de hoy, sino de una fuente secundaria no oficial (Sección 3.2.1). En consecuencia, la equivalencia entre marcos es tan sólida como las fuentes de las que se deriva y queda sujeta a su evolución.

Cobertura parcial de DORA

Por la misma ausencia de un mapeo oficial, el sistema no evalúa DORA en su integridad, sino la parte que admite correspondencia con el catálogo del ENS, modelada como refuerzos (Sección 3.2.1). Quedan fuera del alcance evaluable las pruebas avanzadas de resiliencia (TLPT), el marco de supervisión de los proveedores terceros esenciales y los plazos formales de notificación de incidentes.

Falsos positivos de la búsqueda semántica

La similitud vectorial resultó, por sí sola, un criterio de selección insuficiente: durante el desarrollo se comprobó que controles que comparten el vocabulario genérico de la seguridad, pero evalúan prácticas distintas, superaban el umbral con regularidad, colando en la auditoría preguntas ajenas

al artículo evaluado. Elevar el umbral no era una salida, porque descartaba también candidatos legítimos cuya redacción se aleja de la del texto legislativo. La solución fue arquitectónica (interponer el juez de la Sección 4.8.2 entre la búsqueda y la evaluación), pero tuvo su precio: añadir a la Fase 1 una etapa más lenta y no determinista, lo que conecta esta dificultad con las dos siguientes.

Normalización de identificadores y claves entre subsistemas

La integración entre el *frontend*, el Coordinador y el motor tropezó repetidamente con la heterogeneidad de nomenclaturas: campos en inglés y *camelCase* frente a parámetros en español, el tipo NIS2 viajando como la cadena `"None"` en lugar de un nulo, o diferencias de capitalización en los identificadores de control. Lo traicionero de esta dificultad era su **modo de fallo silencioso**: un desajuste no produce ningún error visible, sino una evaluación que continúa con la normativa afectada desactivada sin avisar; el sistema simplemente exige de menos. De ahí derivan los alias de los modelos Pydantic y la normalización del perfil (Sección 4.12), y la instrumentación `WIRE_DEBUG` (Listado 5.20), creada para poder inspeccionar las cargas reales que cruzan cada frontera.

Latencia de la inferencia y de las conexiones

El tiempo de respuesta fue una fricción constante, con dos fuentes distintas. La primera, el modelo del juez (`qwen2.5:32b`), cuya invocación tarda decenas de segundos: obligó a fijar tiempos de espera amplios (200 segundos) en el Coordinador, a omitir el juez cuando hay dos candidatos o menos y a prever la degradación elegante ante sus fallos (Sección 4.8.2). La segunda, el acceso al grafo en *Neo4j Aura*: el patrón inicial de crear un *driver* por consulta repetía el *handshake* TLS y la autenticación en cada una de las decenas de consultas de una auditoría; su sustitución por el *driver* compartido del Listado 5.4 redujo las baterías de validación de varios minutos a menos de uno.

Deuda técnica en la estructura del código

Por la restricción temporal, el desarrollo priorizó la funcionalidad de extremo a extremo sobre la pulcritud arquitectónica: parte del código presenta funciones extensas y clases con varias responsabilidades. El caso más claro es la clase `AgenteEvaluacionControles`, que reúne la búsqueda semántica (Fase 1), la inferencia del Principio de Máximos (Fase 2), la generación de preguntas, la evaluación de cada control, la puntuación y la traza de

auditoría. Es una limitación de *mantenibilidad*, no de funcionalidad, cuya corrección se plantea en la Sección 7.4.

7.4. Ampliaciones futuras

Módulos de verificación de evidencias

La línea de evolución más relevante consiste en superar el modelo puramente conversacional incorporando módulos específicos de comprobación que validen las evidencias y documentos solicitados al usuario (aportación y análisis de documentación, configuraciones o registros), de modo que el sistema contraste lo declarado con evidencia real en lugar de confiar únicamente en la respuesta textual. Esto convertiría al módulo del ST en una herramienta de verificación asistida.

Persistencia de datos

Dotar al sistema de una capa de persistencia que almacene perfiles, sesiones y veredictos, habilitando el histórico de evaluaciones, la reanudación de sesiones interrumpidas y la trazabilidad a largo plazo.

Actualización y oficialización de los mapeos normativos

Una línea natural de mejora es incorporar las fuentes oficiales a medida que estén disponibles: migrar el mapeo NIS2↔ENS a la nueva versión del PCE-NIS2 cuando el CCN la publique, y sustituir la correspondencia ENS↔DORA de fuente secundaria por un *crosswalk* oficial si llega a existir, ampliando además la cobertura de DORA a los aspectos hoy fuera de alcance, como las pruebas avanzadas de resiliencia (TLPT), la supervisión de proveedores terceros y la notificación de incidentes.

Incorporación de los refuerzos recomendados

Junto a las medidas exigibles por categoría, el ENS contempla una serie de refuerzos que marca como *a considerar*: recomendaciones que la organización puede adoptar pero que no son obligatorias según el nivel del sistema. El grafo de conocimiento ya tiene cargados estos elementos (40 controles y 78 requisitos etiquetados como `OPCIONAL`), pero el motor actual, que activa los requisitos únicamente por las categorías BÁSICA, MEDIA y ALTA, no los incorpora a la evaluación. Una línea natural de mejora es presentarlos en el informe como **recomendaciones**, claramente separadas del veredicto

de cumplimiento obligatorio, de modo que la empresa conozca también las medidas que el ENS sugiere reforzar más allá de lo estrictamente exigido.

Refactorización y modularización

En el plano del código, queda pendiente refactorizar el sistema hacia componentes de **responsabilidad única**, separando el motor de inferencia, la gestión de la conversación y el acceso al grafo en módulos independientes y descomponiendo las funciones más extensas, lo que mejoraría la testabilidad, el mantenimiento y la extensibilidad del código.

7.5. Viabilidad y explotación comercial

Más allá de su valor académico, ***Ciber-AsesorIA* en su conjunto** tiene un **recorrido comercial** que merece argumentarse. El producto que llegaría al mercado no es el módulo del Sistema Técnico aislado, sino la herramienta completa presentada. El asesor que acompaña a una empresa desde la pregunta inicial ("¿qué normativa de ciberseguridad me obliga?") hasta un informe de cumplimiento técnico, encadenando la clasificación regulatoria del perfil, la base de conocimiento normativa del SN y la evaluación técnica conversacional del módulo del ST. El análisis del estado del arte (Sección 3.1.3) mostró que las plataformas del mercado resuelven la *gestión* del cumplimiento, pero ninguna cubre ese recorrido de punta a punta ni realiza la **inferencia normativa** que el módulo del ST aporta al conjunto: determinar, a partir del perfil de la empresa, qué requisitos exige exactamente la confluencia de ENS, NIS2 y DORA mediante el Principio de Máximos. Ese hueco, unido a la presión regulatoria creciente sobre un tejido de pymes sin departamento de cumplimiento (Sección 1.1), define una oportunidad de producto real sobre la que se sustentaría una eventual explotación.

Sostenibilidad de la inferencia

El obstáculo principal para esa explotación es el **coste de la inferencia**, que afecta a los dos subsistemas: tanto la base de conocimiento del SN como el agente y el juez del módulo del ST se apoyan en modelos de lenguaje autoalojados. Durante el desarrollo ese coste ha sido **nulo**, en el caso del módulo del ST gracias a la instancia de Ollama de la Universidad de Granada (Capítulo 5); pero se trata de recursos académicos y temporales, no de una base sostenible para un producto. Una herramienta comercial debe costear sus propios modelos, y ahí surge un **compromiso entre coste y privacidad** decisivo para un producto de ciberseguridad, que maneja respuestas sensibles de la empresa evaluada. La Tabla 7.2 resume las dos vías

Cuadro 7.2: Vías de hospedaje de los modelos de lenguaje para una explotación comercial (precios orientativos, junio de 2026).

Vía	**Coste (orden de magnitud)**	**Dato de la empresa**
API *serverless* por token	Variable: ~0,30–1,20 USD por millón de *tokens* para modelos de la clase empleada [26, 6]	Sale a un tercero (con frecuencia fuera de la UE)
GPU dedicada autoalojada	Fijo: desde ~0,44 USD/h (GPU de 48 GB) hasta ~1,40 USD/h (A100 de 80 GB), del orden de unos cientos de euros al mes en operación continua [23]	Permanece bajo control (ubicable en la UE)

principales.

La vía *serverless* es la más barata a bajo volumen y no exige administrar infraestructura, pero **contradice la garantía de localidad del dato** que el sistema mantiene hoy (Sección 3.1.2 y Capítulo 5): las respuestas del usuario viajarían a un proveedor externo. La GPU autoalojada, preferiblemente en un proveedor europeo o desplegada en la propia infraestructura del cliente, preserva ese control a cambio de un coste fijo. Como la confidencialidad es un argumento de venta central para una herramienta de cumplimiento, la balanza se inclina hacia el autoalojamiento; la decisión definitiva, que debe ponderar coste, volumen de uso y privacidad, queda como un paso previo a la comercialización.

Selección de modelos

El módulo del ST emplea los modelos que la Universidad de Granada puso a disposición (`qwen2.5:14b` para el agente, `qwen2.5:32b` para el juez y `mxbai-embed-large`), y no el resultado de una selección por *benchmark* propio. De cara a la explotación, sería necesario **investigar si modelos más recientes**, por ejemplo familias surgidas con posterioridad como Qwen3 [6], o de menor tamaño mejoran la relación calidad/coste, abaratando la factura de inferencia sin penalizar la precisión del juicio. Esta evaluación queda como línea de trabajo.

Bibliografía

[1] Braue, D. (2025, 28 de mayo). *Cybercrime to cost the world $12.2 trillion annually by 2031*. Cybersecurity Ventures. https://cybersecurityventures.com/official-cybercrime-report-2025/

[2] Centro Criptológico Nacional. (s.f.). *Información sobre el estado de la Guía CCN-STIC 892 Perfil de Cumplimiento Específico para entidades en el ámbito de aplicación del Reglamento de Ejecución (UE) 2024/2690* [Comunicado]. CCN-CERT. Recuperado el 6 de junio de 2026, de https://www.ccn-cert.cni.es/es/seguridad-al-dia/novedades-ccn-cert/13062-informacion-sobre-el-estado-de-la-guia-ccn-stic-892-perfil-de-cumplimiento-especifico-para-entidades-en-el-ambito-de-aplicacion-del-reglamento-de-ejecucion-ue-2024-2690.html

[3] Centro Criptológico Nacional. (2024). *CCN-STIC-892. Perfil de Cumplimiento Específico para organizaciones en el ámbito de aplicación de la Directiva NIS2 (PCE-NIS2)*. NIPO: 083-24-158-2. https://www.ccn-cert.cni.es/es/seguridad-al-dia/novedades-ccn-cert/12945-el-ccn-publica-un-nuevo-perfil-de-cumplimiento-especifico-para-organizaciones-en-el-ambito-de-aplicacion-de-la-directiva-nis2.html

[4] Centro Criptológico Nacional. (2025). *CCN-STIC-808. Esquema Nacional de Seguridad. Verificación del cumplimiento*. NIPO: 083-25-195-8. https://www.ccn-cert.cni.es/es/800-guia-esquema-nacional-de-seguridad/518-ccn-stic-808-verificacion-del-cumplimiento-de-las-medidas-en-el-ens/file.html

[5] Chung, J., Ko, R., Yoo, W., Onizuka, M., Kim, S., Kim, T.-W., & Shin, W.-Y. (2025). *GraphCompliance: Aligning policy and context graphs for LLM-based regulatory compliance* (arXiv:2510.26309). arXiv. https://arxiv.org/abs/2510.26309

[6] DeepInfra. (s.f.). *Deep Infra: Pricing y catálogo de modelos de ge-*

neración de texto. Recuperado el 8 de junio de 2026, de https://deepinfra.com/models/text-generation

[7] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2024). *From local to global: A Graph RAG approach to query-focused summarization* (arXiv:2404.16130). arXiv. https://arxiv.org/abs/2404.16130

[8] ENISA. (2025). *ENISA threat landscape 2025*. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2025

[9] Ershov, V. (2023). *A case study for compliance as code with graphs and language models: Public release of the Regulatory Knowledge Graph* (arXiv:2302.01842). arXiv. https://arxiv.org/abs/2302.01842

[10] Formalize. (s.f.). *Software DORA*. Recuperado el 5 de junio de 2026, de https://formalize.com/es/dora

[11] GlobalSuite Solutions. (s.f.). *Cumplimiento NIS2: plataforma integral para tu ciberresiliencia*. Recuperado el 5 de junio de 2026, de https://www.globalsuitesolutions.com/es/cumplimiento-nis2/

[12] Gobierno de España. (2026, 26 de mayo). *Proyecto de Ley Orgánica para el buen uso y la gobernanza de la inteligencia artificial* [Presentación]. Consejo de Ministros. https://www.lamoncloa.gob.es/consejodeministros/resumenes/Documents/2026/260526-presentacion-proyecto-ley-ia.pdf

[13] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. En *Advances in Neural Information Processing Systems* (Vol. 33, pp. 9459–9474). https://arxiv.org/abs/2005.11401

[14] Neo4j. (s.f.). *Free graph database: AuraDB Free*. Recuperado el 5 de junio de 2026, de https://neo4j.com/free-graph-database/

[15] OneTrust. (s.f.). *Navegando el Reglamento sobre resiliencia operativa digital (DORA) con OneTrust*. Recuperado el 5 de junio de 2026, de https://www.onetrust.com/es/blog/navigating-the-digital-operational-resilience-act-dora-with-onetrust/

[16] Parlamento Europeo y Consejo de la Unión Europea. (2022a). *Reglamento (UE) 2022/2554 del Parlamento Europeo y del Consejo, de 14 de diciembre de 2022, sobre la resiliencia operativa digital del sec-*

tor financiero (DORA). Diario Oficial de la Unión Europea, L 333. https://eur-lex.europa.eu/eli/reg/2022/2554/oj

[17] Parlamento Europeo y Consejo de la Unión Europea. (2022b). *Directiva (UE) 2022/2555 del Parlamento Europeo y del Consejo, de 14 de diciembre de 2022, relativa a las medidas destinadas a garantizar un elevado nivel común de ciberseguridad en toda la Unión (Directiva NIS2)*. Diario Oficial de la Unión Europea, L 333. https://eur-lex.europa.eu/eli/dir/2022/2555/oj

[18] Parlamento Europeo y Consejo de la Unión Europea. (2024). *Reglamento (UE) 2024/1689 del Parlamento Europeo y del Consejo, de 13 de junio de 2024, por el que se establecen normas armonizadas en materia de inteligencia artificial (Reglamento de Inteligencia Artificial)*. Diario Oficial de la Unión Europea, serie L. https://eur-lex.europa.eu/eli/reg/2024/1689/oj

[19] Proactivanet. (s.f.). *Cumplimiento ENS* [ProactivaENS v3]. Recuperado el 10 de abril de 2026, de https://www.proactivanet.com/compliance-ens/

[20] Real Decreto 311/2022, de 3 de mayo, por el que se regula el Esquema Nacional de Seguridad. *Boletín Oficial del Estado*, núm. 106, de 4 de mayo de 2022, pág. 61715. https://www.boe.es/buscar/act.php?id=BOE-A-2022-7191

[21] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[22] Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph databases: New opportunities for connected data* (2.ª ed.). O'Reilly Media. https://graphdatabases.com/

[23] RunPod. (s.f.). *RunPod: Pricing (GPU cloud)*. Recuperado el 8 de junio de 2026, de https://www.runpod.io/pricing

[24] Seguridad Social. (s.f.). *Cuota reducida (tarifa plana) para nuevos autónomos*. Ministerio de Inclusión, Seguridad Social y Migraciones. Recuperado el 22 de junio de 2026, de https://www.seg-social.es/

[25] TigerData. (s.f.). *Finding the best open source embedding model for RAG*. Recuperado el 6 de junio de 2026, de https://www.tigerd

ata.com/blog/finding-the-best-open-source-embedding-model-for-rag

[26] Together AI. (s.f.). *Together AI: Pricing*. Recuperado el 8 de junio de 2026, de https://www.together.ai/pricing

[27] Ulises GRC. (s.f.). *Ulises GRC: la herramienta definitiva para gestionar todos los riesgos de su compañía*. Recuperado el 5 de junio de 2026, de https://ulisesgrc.com/

[28] Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., ... Qiu, Z. (2024). *Qwen2.5 technical report* (arXiv:2412.15115). arXiv. https://arxiv.org/abs/2412.15115

Apéndice A

Esquema del grafo Neo4j ST y consultas *Cypher*

El modelo del grafo de conocimiento técnico (sus cinco tipos de nodo y cinco relaciones) se describe en el Capítulo 4 (Figura 4.3 y Tablas 4.2, 4.3 y 4.1). Este anexo no lo repite: recoge la **correspondencia entre las reglas en JSON y las propiedades del grafo** y las **consultas *Cypher* completas** que el motor de inferencia ejecuta, mostradas en el cuerpo solo de forma abreviada.

A.1. Correspondencia entre el JSON de reglas y las propiedades del grafo

Las reglas del PCE-NIS2 (Anexo C) se almacenan en `reglas_pce_nis2.json` con nombres descriptivos, y el cargador `st_loader_nis2.py` las vuelca, con nombres más cortos, como propiedades de la relación `CUMPLE_NIS2`, que son las que lee la consulta del Principio de Máximos. La Tabla A.1 recoge esa correspondencia.

A.2. Consultas de inspección del grafo

El recuento de nodos y relaciones por tipo (Tabla 4.1) se obtiene con:

Y el subgrafo de un control concreto con todas sus relaciones (como el de la Figura 4.4) se recupera así:

Cuadro A.1: Correspondencia entre los campos de `reglas_pce_nis2.json` y las propiedades de la relación `CUMPLE_NIS2`.

Campo en el JSON	**Propiedad en `CUMPLE_NIS2`**
`esencial_minima_global`	`esencial_min_global`
`importante_minima_global`	`importante_min_global`
`esencial_minimo_nivel`	`esencial_min_nivel`
`importante_minimo_nivel`	`importante_min_nivel`
`esencial_requiere_refuerzos`	`esencial_req`
`importante_requiere_refuerzos`	`importante_req`
`importante_excluye_refuerzos`	`importante_excl`
`excluido_pce_nis2`	`excluido_pce_nis2`
`excluido_esenciales`	`excluido_esenciales`
`excluido_importantes`	`excluido_importantes`
`sectores_especificos`	`sectores_especificos`
`condiciones_nis2`	`condiciones_nis2`

Listado A.1: Recuento de nodos y relaciones por tipo.

```
// Nodos por etiqueta
MATCH (n) RETURN labels(n)[0] AS tipo, count(*) AS total ORDER BY total DESC;

// Relaciones por tipo
MATCH ()-[r]->() RETURN type(r) AS tipo, count(*) AS total ORDER BY total DESC;
```

Listado A.2: Subgrafo de un control y sus relaciones.

```
MATCH p = (m:MarcoENS)-[:AGRUPA_CONTROL]->(c:ControlENS {id_control: "Org.1"})
          -[:EVALUA_MEDIANTE|CUMPLE_NIS2|CUMPLE_DORA|REFUERZA_A]-()
RETURN p;
```

Listado A.3: Búsqueda de controles por similitud coseno (Fase 1).

```
MATCH (c:ControlENS)
WHERE c.vector IS NOT NULL
  AND NOT EXISTS { (c)-[:REFUERZA_A]->(:ControlENS) }
  // + filtro por normativa del perfil cuando ENS no aplica
  //   (EXISTS CUMPLE_DORA y/o CUMPLE_NIS2 segun el perfil)
WITH c, gds.similarity.cosine(c.vector, $embedding_query) AS similarity
WHERE similarity > 0.65
ORDER BY similarity DESC
RETURN DISTINCT c.id_control AS id_control, c.titulo AS titulo, similarity;
```

A.3. Fase 1: búsqueda por similitud

La recuperación de controles candidatos por similitud coseno (Sección 4.8.1) es:

A.4. Fase 2: Principio de Máximos (consulta completa)

La consulta que resuelve, para un control, qué requisitos auditar y a qué nivel, integrando ENS, NIS2 y DORA, se muestra abreviada en la Sección 4.8.3. Se reproduce aquí completa (función `_buscar_requisitos_en_st`). Recibe como parámetros el identificador del control, la categoría global de la empresa, el diccionario `$dims_empresa` con su nivel en cada dimensión CITAD, el tipo de entidad NIS2, los *flags* de DORA y ENS, y el diccionario de pesos (Tabla 4.4). La consulta se organiza en **cinco bloques**, señalados con comentarios `// 1` – `// 5` en el propio código:

1. **Localizar el control.** Recupera el `ControlENS` por su identificador (sin distinguir mayúsculas).
2. **Normativas y nivel ganador.** Sigue las relaciones `CUMPLE_NIS2` y `CUMPLE_DORA` y determina primero si cada normativa **aplica de verdad** (`nis2_valido` / `dora_valido`): la NIS2 se descarta si la relación marca una exclusión (`excluido_pce_nis2` / `excluido_esenciales` / `excluido_importantes`) o si la entidad no está sujeta. A continuación calcula tres pesos: el del riesgo propio de la empresa, tomado como el máximo de sus niveles en las dimensiones que el control afecta (`p_empresa`, Sección 4.8.3), el suelo global de la NIS2 (`p_global`) y el suelo dimensional de la NIS2 (`p_dim`); y toma el **máximo**, el `peso_ganador`.
3. **Requisitos base.** Selecciona los `RequisitoENS` del control (relación `EVALUA_MEDIANTE`) cuya categoría de aplicación queda alcanzada por

el `peso_ganador`, y anota a cada uno sus **orígenes** (ENS, NIS2 y/o DORA), conservando solo los que tengan al menos uno.

4. **Refuerzos.** Incorpora los controles enlazados por `REFUERZA_A` que active el nivel ganador **o** que exijan explícitamente la NIS2 (`esencial_req` / `importante_req`) o DORA (`requiere_refuerzos`), descartando los que la NIS2 perdone (`importante_excl`); de cada refuerzo recoge sus requisitos con sus orígenes.

5. **Consolidación.** Devuelve la unión de los requisitos base y los de refuerzo.

Listado A.4: Consulta Cypher completa del Principio de Máximos (Fase 2).

```
// 1. Localizar el control
MATCH (c:ControlENS) WHERE toLower(c.id_control) = toLower($id_control)

// 2. Evaluar normativas y pesos (cálculo del nivel ganador)
OPTIONAL MATCH (c)-[rel_n:CUMPLE_NIS2]->(n:ArticuloNIS2)
OPTIONAL MATCH (c)-[rel_d:CUMPLE_DORA]->(d:ArticuloDORA)

WITH c, n, d, rel_n, rel_d,
     (CASE
        WHEN rel_n.excluido_pce_nis2 = true THEN false
        WHEN $tipo_nis2 = 'ESENCIAL'   AND rel_n.excluido_esenciales  = true THEN
     false
        WHEN $tipo_nis2 = 'IMPORTANTE' AND rel_n.excluido_importantes = true THEN
     false
        WHEN $tipo_nis2 = 'NINGUNO' THEN false
        ELSE (n IS NOT NULL)
     END) AS nis2_valido,
     ($es_dora = true AND d IS NOT NULL) AS dora_valido

WITH c, n, d, rel_n, rel_d, nis2_valido, dora_valido,
     $pesos[toUpper(coalesce(
        CASE WHEN nis2_valido
             THEN (CASE WHEN $tipo_nis2 = 'ESENCIAL'
                        THEN rel_n.esencial_min_global ELSE rel_n.
     importante_min_global END)
        END, "NINGUNO"))] AS p_global,
     $pesos[toUpper(coalesce(
        CASE WHEN nis2_valido
             THEN (CASE WHEN $tipo_nis2 = 'ESENCIAL'
                        THEN rel_n.esencial_min_nivel ELSE rel_n.
     importante_min_nivel END)
        END, "NINGUNO"))] AS p_dim,
     // Riesgo propio POR DIMENSION (ENS Anexo II / 892): max del nivel de la
     // empresa en las dimensiones del control (siempre, no depende de aplica_ens)
     reduce(m = 0,
            dd IN (CASE WHEN c.dimensiones IS NULL OR size(c.dimensiones) = 0
                        THEN ['C','I','T','A','D'] ELSE c.dimensiones END) |
            CASE WHEN $pesos[toUpper(coalesce($dims_empresa[dd], 'NINGUNO'))] > m
                 THEN $pesos[toUpper(coalesce($dims_empresa[dd], 'NINGUNO'))]
                 ELSE m END) AS p_empresa

WITH c, n, d, rel_n, rel_d, nis2_valido, dora_valido, p_empresa,
     (CASE
        WHEN p_global >= p_dim AND p_global >= p_empresa THEN p_global
```

```
        WHEN p_dim   >= p_global AND p_dim   >= p_empresa THEN p_dim
        ELSE p_empresa
     END) AS peso_ganador

// 3. Requisitos base
CALL (c, peso_ganador, nis2_valido, dora_valido, n, d, p_empresa) {
    OPTIONAL MATCH (c)-[:EVALUA_MEDIANTE]->(rb:RequisitoENS)
    WHERE (peso_ganador >= 1 AND "BÁSICA" IN rb.categorias_aplicables)
       OR (peso_ganador >= 2 AND "MEDIA"  IN rb.categorias_aplicables)
       OR (peso_ganador >= 3 AND "ALTA"   IN rb.categorias_aplicables)
    WITH rb, c,
         (CASE WHEN nis2_valido THEN ["NIS2: " + n.id_articulo] ELSE [] END) +
         (CASE WHEN dora_valido THEN ["DORA: " + d.id_dora]     ELSE [] END) +
         (CASE WHEN $aplica_ens = true AND p_empresa >= $pesos[rb.
     categorias_aplicables[0]]
               THEN ["ENS Base (" + $cat_global + ")"] ELSE [] END) AS s
    WHERE rb IS NOT NULL AND size(s) > 0
    RETURN collect({desc: rb.descripcion, tipo: rb.tipo, es_esencial: rb.es_esencial
     ,
                    pertenece_a: c.id_control, origenes: s}) AS base_reqs
}

// 4. Refuerzos (inyeccion segun nivel, NIS2 y DORA; exclusiones)
CALL (c, peso_ganador, nis2_valido, dora_valido, n, d, rel_n, rel_d) {
    OPTIONAL MATCH (ref:ControlENS)-[:REFUERZA_A]->(c)
    WITH ref, n, d, nis2_valido, dora_valido, peso_ganador,
         (CASE WHEN $tipo_nis2 = 'ESENCIAL'   THEN coalesce(rel_n.esencial_req, [])
               WHEN $tipo_nis2 = 'IMPORTANTE' THEN coalesce(rel_n.importante_req,
     [])
               ELSE [] END) AS n_add,
         (CASE WHEN $tipo_nis2 = 'IMPORTANTE'
               THEN coalesce(rel_n.importante_excl, []) ELSE [] END) AS n_del,
         (CASE WHEN dora_valido THEN coalesce(rel_d.requiere_refuerzos, []) ELSE []
     END) AS d_add
    WHERE (
        (peso_ganador >= 2 AND "MEDIA" IN ref.categorias_aplicables)
        OR (peso_ganador >= 3 AND "ALTA"  IN ref.categorias_aplicables)
        OR ANY(r IN n_add WHERE toLower(ref.id_control) ENDS WITH "." + toLower(r))
        OR ANY(r IN d_add WHERE toLower(ref.id_control) ENDS WITH "." + toLower(r))
    )
    AND NOT ANY(excl IN n_del WHERE toLower(ref.id_control) ENDS WITH "." + toLower(
     excl))
    OPTIONAL MATCH (ref)-[:EVALUA_MEDIANTE]->(rr:RequisitoENS)
    WITH rr, ref,
         (CASE WHEN ANY(r IN n_add WHERE toLower(ref.id_control) ENDS WITH "." +
     toLower(r))
               THEN ["NIS2: " + n.id_articulo] ELSE [] END) +
         (CASE WHEN ANY(r IN d_add WHERE toLower(ref.id_control) ENDS WITH "." +
     toLower(r))
               THEN ["DORA: " + d.id_dora] ELSE [] END) +
         (CASE WHEN $aplica_ens = true AND peso_ganador >= $pesos[ref.
     categorias_aplicables[0]]
               THEN ["ENS Ref: " + ref.id_control] ELSE [] END) AS s
    WHERE rr IS NOT NULL AND size(s) > 0
    RETURN collect({desc: rr.descripcion, tipo: rr.tipo, es_esencial: rr.es_esencial
     ,
                    pertenece_a: ref.id_control, origenes: s}) AS ref_reqs
}

// 5. Consolidacion
RETURN c.id_control AS id_control, c.titulo AS titulo,
       (base_reqs + ref_reqs) AS requisitos,
```

```
94          (CASE WHEN nis2_valido THEN rel_n.condiciones_nis2 ELSE null END)
95              AS avisos_nis2;
```

Esta consulta puede devolver un mismo requisito **más de una vez**: por ejemplo, cuando la misma medida llega por el control base y, además, por un refuerzo, o cuando la justifican a la vez varias normativas; cada aparición trae su propia etiqueta de origen. Por eso, sobre el resultado, el código Python **deduplica los requisitos usando su descripción como clave**: conserva una sola copia de cada uno y **fusiona sus orígenes** (la lista de normativas, ENS, NIS2 y DORA, que lo exigen), de modo que un requisito que procede de varias vías no se pregunta dos veces y queda anotado con todas ellas. Ese paso se muestra en el Listado 5.8 del Capítulo 5.

Junto a los requisitos, la consulta devuelve un último campo, `avisos_nis2`, que, cuando la NIS2 resulta aplicable, transporta las `condiciones_nis2` de la relación: **notas de auditor** y **plazos de notificación** (Anexo C). Se trata de anotaciones **informativas** que se trasladan al evaluador como contexto adicional y **no alteran qué requisitos resultan exigibles**; por eso viajan al margen de las dos listas de requisitos y no intervienen en el cálculo del nivel ganador.

Apéndice B

Ejemplo de *audit trail* y de conversación de evaluación

Este anexo ilustra el funcionamiento del sistema completo (*frontend*, Coordinador, SN y módulo del ST) sobre una evaluación real de extremo a extremo, desde el perfil de la empresa hasta el informe final.

La auditoría completa se aporta como **demostración en vídeo**. Dado que la ejecución íntegra supera los diez minutos, la grabación se ha recortado para mostrar los tramos representativos del recorrido. Puede consultarse en el siguiente enlace:

`https://drive.google.com/file/d/1fda5OxRT8-QzjLk236pUyaDy8WqkhWik/view?usp=drive_link`

A modo de complemento al vídeo, las capturas que siguen recogen, sobre la misma ejecución, los momentos representativos del recorrido: tanto los **registros (*logs*) de los servicios**, que evidencian el funcionamiento interno del motor, como las **vistas del *frontend*** que recibe el usuario (la conversación y el informe final).

B.1. Punto de partida: perfil de la empresa y controles aplicables

La evaluación arranca con el perfil que el usuario construye en el *frontend*. La Figura B.1 muestra el arranque del módulo del ST y la creación de la sesión en el Coordinador con ese perfil. El módulo del ST levanta con los modelos `qwen2.5:14b` (agente) y `qwen2.5:32b` (juez) servidos por la instancia de Ollama de la UGR (`:11435`) y conectado a su grafo *Neo4j Aura*.

Figura B.1: Arranque del módulo del ST y creación de la sesión: el perfil de la empresa (ENS, categoría global MEDIA y dimensiones CITAD) y los 53 controles legislativos que el SN devuelve para ese perfil.

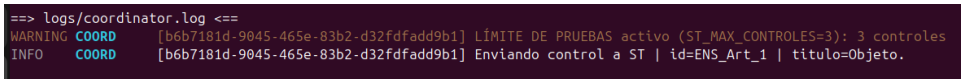

Figura B.2: El Coordinador acota la ejecución a tres controles legislativos (`ST_MAX_CONTROLES=3`) y envía el primero, `ENS_Art_1`, al módulo del ST.

Para esta ejecución, la empresa pertenece al sector de la energía y es de tamaño pequeño; le aplica **únicamente el ENS** (`applies_ens=True`, sin DORA ni NIS2), con **categoría global MEDIA** y dimensiones de seguridad (CITAD) C = MEDIA, I = BÁSICA, T = MEDIA, A = MEDIA y D = MEDIA. A partir de ese perfil, el SN, tratado como caja negra (Anexo D), consulta su grafo y devuelve los **53 controles legislativos** del ENS aplicables, que son los que el módulo del ST evaluará uno a uno.

B.2. Acotación de la demostración a tres controles

Evaluar los 53 controles legislativos de extremo a extremo es un proceso largo, ya que cada uno desencadena una conversación completa con el módulo del ST. Para que esta demostración resulte abarcable, la ejecución se ha limitado a **3 controles legislativos** mediante el parámetro `ST_MAX_CONTROLES`. Como recoge la Figura B.2, el Coordinador deja constancia de esa acotación con un aviso (`LÍMITE DE PRUEBAS activo`) y, a continuación, envía el primer control al ST: `ENS_Art_1` (*Objeto*). Se trata de un límite de la **demostración**, no del sistema: sin él, el recorrido procesaría los 53 controles de igual modo.

Figura B.3: Fase 1 para `ENS_Art_1`: la búsqueda por similitud coseno preselecciona 30 controles técnicos candidatos y el juez (`qwen2.5:32b`) los reduce a los 2 pertinentes (`Art.41` y `Art.38`).

B.3. Fase 1: descubrimiento semántico y selección del juez

Recibido el control `ENS_Art_1` (*Objeto*), el módulo del ST ejecuta el turno de preparación, que comprende las dos fases del motor de inferencia (Sección 4.8). La Figura B.3 recoge la **Fase 1**. El texto del control legislativo se vectoriza con `mxbai-embed-large` y, mediante similitud coseno sobre el grafo, se pre-seleccionan los **30 controles técnicos** más próximos; el filtro opera en modo *universal*, pues al perfil solo le aplica el ENS. La lista la encabeza `Art.40` (similitud 0,725).

Sobre esos 30 candidatos interviene el **juez** (`qwen2.5:32b`), que descarta el ruido semántico y reduce la selección a los **2 controles técnicos realmente pertinentes**, `Art.41` y `Art.38`, dejando registrada la razón de su elección. Es la salvaguarda que evita auditar controles que solo se parecen superficialmente al artículo.

B.4. Fase 2 e inicio de la conversación

Con los dos controles base elegidos por el juez, la **Fase 2** resuelve qué requisitos hay que auditar. Para `Art.41` y `Art.38`, el motor aplica el **Principio de Máximos** (Sección 4.8.3) con los parámetros del perfil (categoría MEDIA, dimensiones CITAD, sin NIS2 ni DORA, solo ENS), de donde resultan **4 requisitos** para `Art.41` y **2** para `Art.38` (Figura B.4). Cerrada la lista de requisitos, el Coordinador inicia la sesión (3 controles, ENS) y el agente (`qwen2.5:14b`) formula la primera pregunta, correspondiente al

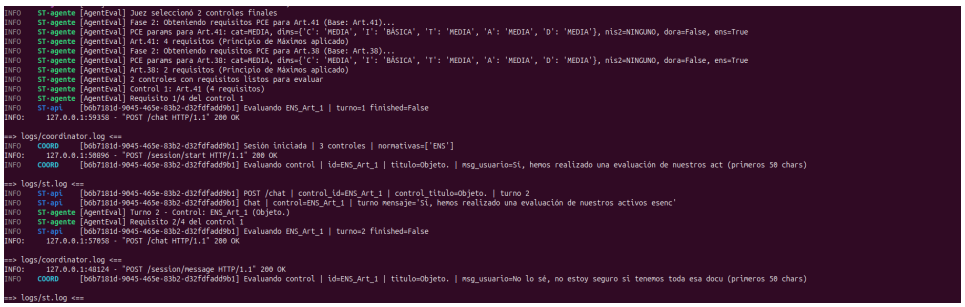

Figura B.4: Fase 2 para `ENS_Art_1`: el Principio de Máximos obtiene 4 requisitos de `Art.41` y 2 de `Art.38`; arranca entonces la conversación, requisito a requisito.

Requisito 1/4 de `Art.41`.

La Figura B.5 muestra esa primera pregunta tal como la recibe el usuario en el *frontend*: el agente pregunta si se han evaluado los activos esenciales en las cinco dimensiones de seguridad y enumera las evidencias que lo acreditarían. Para conducir el recorrido completo sin teclear cada respuesta a mano, la contestación la genera el **usuario simulado** (Capítulo 6), que en este turno responde afirmativamente. El intercambio prosigue requisito a requisito (en el registro, el turno 2 ya atiende el *Requisito 2/4*).

B.5. Veredicto técnico y caché de veredictos

A medida que se recaban las respuestas, el agente emite el veredicto de cada control técnico y lo **guarda en la caché de veredictos de la sesión**. La Figura B.6 lo muestra ya dentro del segundo control legislativo, `ENS_Art_10` (*Vigilancia continua y reevaluación periódica*): al evaluar el control técnico `Op.mon.3`, el agente extrae el JSON `{cumple, razón}` con `cumple=True` y lo registra (`Cache guardado: Op.mon.3 ->cumple=True`) antes de pasar al siguiente control base, `Org.3`. Si ese mismo control técnico reaparece como pertinente para otro artículo, su veredicto se reutiliza sin volver a preguntar al usuario (Sección 4.9.3).

B.6. Evaluación final y persistencia en el *audit trail*

Una vez recabados los requisitos de los tres controles legislativos, el agente emite el último veredicto técnico y cierra la sesión. La Figura B.7 muestra ese cierre: tras evaluar `Op.exp.2` (`cumple=True`), el registro marca `FIN:`

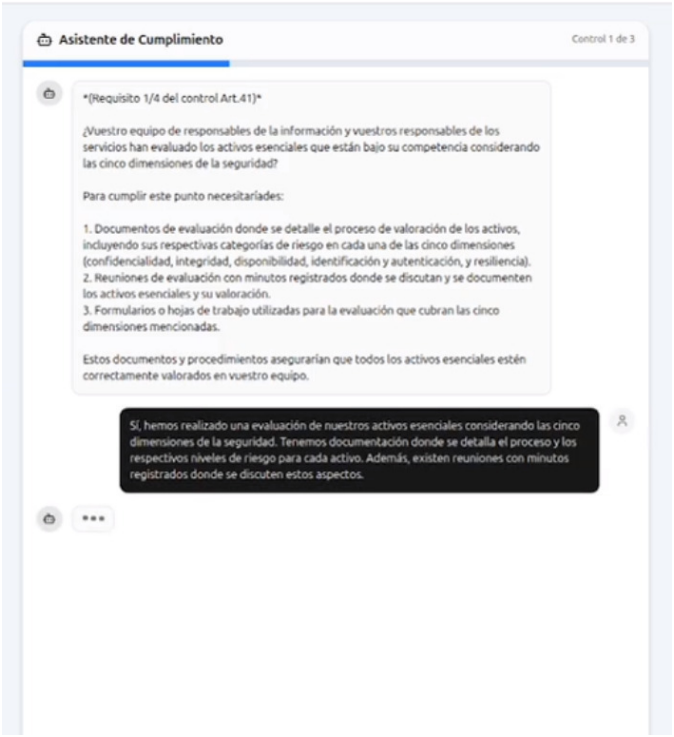

Figura B.5: Primera pregunta de la evaluación (*Requisito 1/4* de `Art.41`) vista en el *frontend*, con la respuesta generada por el usuario simulado.

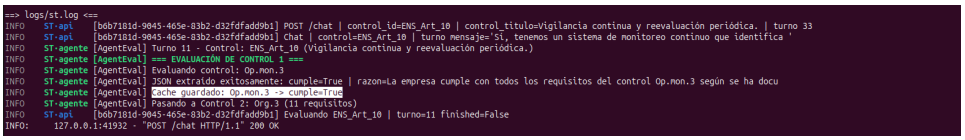

Figura B.6: Evaluación del control técnico `Op.mon.3` (`cumple=True`) y su registro en la caché de veredictos de la sesión, que evita reevaluarlo si reaparece bajo otro artículo.

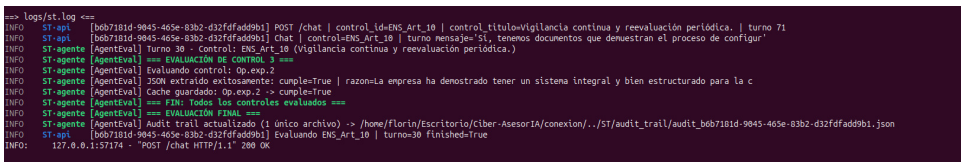

Figura B.7: Cierre de la evaluación: todos los controles evaluados y volcado de la traza de auditoría al fichero JSON único de la sesión.

`Todos los controles evaluados` y `EVALUACIÓN FINAL`, y la traza de auditoría se persiste en un **único fichero JSON por sesión** (`ST/audit_trail/audit_<session_id>.json`).

Las Figuras B.8 y B.9 reproducen ese *audit trail* completo de la ejecución (cuya estructura se describe en el Listado 5.13). Por cada control legislativo evaluado se registra: el **control legislativo** (`id`, título y descripción), el **resultado del ST** (`resultado_st`: `cumple`, `puntuacion` 1–10 y `razon`) y el detalle de los **controles técnicos** auditados, cada uno con su veredicto (`cumple`), la `razon_llm` del modelo, la `similitud` que motivó su selección y sus `origenes` normativos.

Sobre los tres controles de esta sesión se aprecia la **puntuación determinista** en acción: `ENS_Art_1` queda en `cumple=false` con puntuación 6 (de sus dos controles técnicos, `Art.41` cumple y `Art.38` no), mientras que `ENS_Art_10` y `ENS_Art_11` alcanzan `cumple=true` con puntuación 10. También queda registrada la **reutilización de la caché**: en `ENS_Art_11`, el control técnico `Art.41` arrastra la marca `[Caché sesión -- evaluado para 'Objeto.']`, con `similitud: null` y `origenes: []`, porque su veredicto se reutilizó del primer control sin volver a interrogar al usuario.

B.7. Informe de cumplimiento final

Cerrada la evaluación técnica, el Coordinador remite los veredictos de los tres controles al SN, que elabora el **informe de cumplimiento** que el usuario recibe en el *frontend* (Anexo D). La Figura B.10 lo recoge. Coherente con el *audit trail*, el informe emite un **veredicto global de NO CUMPLE** con un **66,7 % de controles superados** (2 de 3) sobre un perfil `ENS MEDIA`, e incluye el **perfil de cumplimiento CITAD** (nivel esperado frente a nivel evaluado en las cinco dimensiones), las brechas detectadas, un plan de acción y un resumen ejecutivo con las actuaciones priorizadas. Con ello se completa el recorrido de extremo a extremo, del perfil de la empresa al informe final.

```
{
  "session_id": "b6b7181d-9045-465e-83b2-d32fdfadd9b1",
  "started_at": "2026-06-22T21:39:21.072371",
  "evaluaciones": [
    {
      "timestamp": "2026-06-22T21:39:21.072325",
      "control_legislativo": {
        "id": "ENS_Art_1",
        "titulo": "Objeto.",
        "descripcion": "[ENS] 1. Este real decreto tiene por objeto regular el Esquema Nacional de Seguridad (en adelante, ENS), establecido en el artículo 15
      },
      "resultado_st": {
        "cumple": false,
        "puntuacion": 6,
        "razon": "Art.38: No se ha proporcionado evidencia clara de que las Declaraciones y Certificaciones de Conformidad con el ENS estén publicadas en las
        "sin_control_tecnico": false
      },
      "controles_tecnicos": [
        {
          "control_id": "Art.41",
          "cumple": true,
          "razon_llm": "La empresa ha proporcionado evidencia de que sus responsables de información y servicios han evaluado los activos esenciales considera
          "similitud": 0.6977992126185106,
          "origenes": [
            "ENS Base (MEDIA)"
          ]
        },
        {
          "control_id": "Art.38",
          "cumple": false,
          "razon_llm": "No se ha proporcionado evidencia clara de que las Declaraciones y Certificaciones de Conformidad con el ENS estén publicadas en las pá
          "similitud": 0.7035304369127445,
          "origenes": [
            "ENS Base (MEDIA)"
          ]
        }
      ]
    },
    {
      "timestamp": "2026-06-22T21:44:56.573423",
      "control_legislativo": {
        "id": "ENS_Art_10",
        "titulo": "Vigilancia continua y reevaluación periódica.",
        "descripcion": "[ENS] 1. La vigilancia continua permitirá la detección de actividades o comportamientos anómalos y su oportuna respuesta.\n2. La evalu
      },
      "resultado_st": {
        "cumple": true,
        "puntuacion": 10,
        "razon": "Todos los controles técnicos evaluados cumplen los requisitos.",
```


Figura B.8: Traza de auditoría de la sesión (primera parte): cabecera y evaluación de `ENS_Art_1`, con el resultado del ST y sus controles técnicos (`Art.41` y `Art.38`).

```
      "controles_tecnicos": [
        {
        }
      ]
    },
    {
      "timestamp": "2026-06-22T21:45:24.540155",
      "control_legislativo": {
        "id": "ENS_Art_11",
        "titulo": "Diferenciación de responsabilidades.",
        "descripcion": "[ENS] 1. En los sistemas de información se diferenciará el responsable de la información, el responsable del servicio, el responsable de la seguri
      },
      "resultado_st": {
        "cumple": true,
        "puntuacion": 10,
        "razon": "Todos los controles técnicos evaluados cumplen los requisitos.",
        "sin_control_tecnico": false
      },
      "controles_tecnicos": [
        {
          "control_id": "Art.41",
          "cumple": true,
          "razon_llm": "[Caché sesión — evaluado para 'Objeto.'] La empresa ha proporcionado evidencia de que sus responsables de información y servicios han evaluado los
          "similitud": null,
          "origenes": []
        },
        {
          "control_id": "Art.40",
          "cumple": true,
          "razon_llm": "La empresa cumple con el requisito del control Art.40 ya que dispone de un informe detallado donde se reflejan las valoraciones realizadas sobre
          "similitud": 0.6732555839590287,
          "origenes": [
            "ENS Base (MEDIA)"
          ]
        }
      ]
    }
  ],
  "last_updated": "2026-06-22T21:45:24.540333"
}
```


Figura B.9: Traza de auditoría de la sesión (segunda parte): evaluación de `ENS_Art_11`, donde `Art.41` se reutiliza de la caché (`similitud: null`).

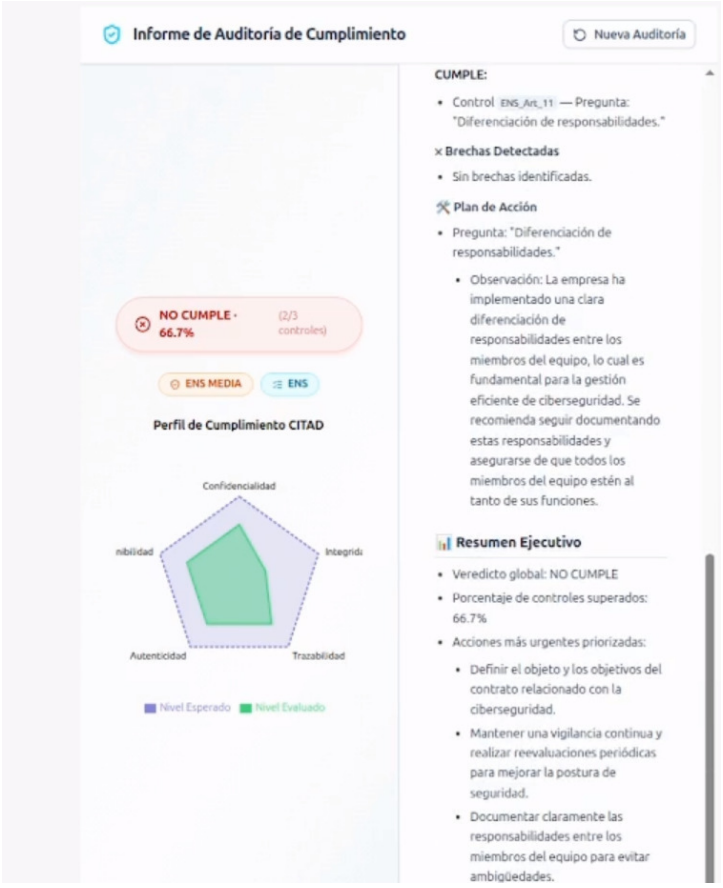

Figura B.10: Informe de cumplimiento que el SN elabora a partir de los veredictos técnicos y que el usuario recibe en el *frontend*: veredicto global, porcentaje de controles superados, perfil CITAD, brechas, plan de acción y resumen ejecutivo.

Apéndice C

La proyección de NIS2 y DORA sobre el ENS: fuentes y modelado

Este anexo documenta el **origen normativo** de los suelos, refuerzos y exclusiones que el Principio de Máximos aplica sobre los controles del ENS (Sección 4.8.3), así como la forma en que se han codificado en el grafo. Los pesos de nivel empleados en el cálculo se recogen en la Tabla 4.4 del Capítulo 4; aquí se detallan las fuentes de las que se derivan las reglas de NIS2 y DORA.

C.1. NIS2: el Perfil de Cumplimiento Específico PCE-NIS2 (CCN-STIC-892)

La proyección de la NIS2 sobre el ENS procede del Perfil de Cumplimiento Específico publicado por el Centro Criptológico Nacional, la guía **CCN-STIC-892** [3]. Según su apartado 6.3, al PCE-NIS2 le aplican **72 de las 73 medidas** de seguridad del Anexo II del RD 311/2022 (17 procedentes de categoría ALTA, 30 de MEDIA y 25 de BÁSICA); algunas se potencian con refuerzos y otras se excluyen según la entidad sea esencial o importante. El grafo de este trabajo modela exactamente esas **72 medidas**, lo que confirma la cobertura completa del perfil.[1]

El núcleo del perfil es su **Declaración de Aplicabilidad** (apartado 6.2

[1] La medida del Anexo II que el PCE-NIS2 no proyecta sobre la NIS2 es `[mp.info.2] Calificación de la información`, excluida de su Declaración de Aplicabilidad [3] (ap. 6.2); el grafo la conserva como control del ENS, sin exigencia alguna derivada de la NIS2.

6.2 Detalle de la Declaración de Aplicabilidad

Dimensiones Afectadas	CAT B	CAT M	CAT A	Medida	Importantes	Esenciales	Aplicación[1]	Ref
Categoría	aplica	aplica	aplica	org.1	SI	SI	BÁSICA	
Categoría	aplica	aplica	aplica	org.2	SI	SI	BÁSICA	
Categoría	aplica	aplica	aplica	org.3	SI	SI	BÁSICA	
Categoría	aplica	aplica	aplica	org.4	SI	SI	BÁSICA	
Categoría	aplica	+ R1	+ R2	op.pl.1	SI *	SI	MEDIA	71
Categoría	aplica	+ R1	+ R1 + R2 + R3	op.pl.2	SI	SI	MEDIA	
Categoría	aplica	aplica	aplica	op.pl.3	SI	SI	BÁSICA	
D	aplica	+ R1	+ R1	op.pl.4	SI	SI	Bajo	
Categoría	n.a.	aplica	aplica	op.pl.5	SI	SI	MEDIA (+R1+R2)	7.2
T A	aplica	+ R1	+ R1	op.acc.1	SI	SI	Medio	
C I T A	aplica	aplica	+ R1	op.acc.2	SI	SI	Alto	
C I T A	n.a.	aplica	+ R1	op.acc.3	SI	SI	Medio	
C I T A	aplica	aplica	aplica	op.acc.4	SI	SI	Bajo	
C I T A	+ [R1 o R2 o R3 o R4]	+ [R2 o R3 o R4] + R5	+ [R2 o R3 o R4] + R5	op.acc.5	SI *	SI	Medio	7.3
C I T A	+ [R1 o R2 o R3 o R4] + R8 + R9	+ [R1 o R2 o R3 o R4] + R5 + R8 + R9	+ [R1 o R2 o R3 o R4] + R5 + R6 + R7 + R8 + R9	op.acc.6	SI *	SI	Medio	7.4
Categoría	aplica	aplica	aplica	op.exp.1	SI*	SI	BÁSICA (+R1+R2+R3+R4)	7.5
Categoría	aplica	aplica	aplica	op.exp.2	SI	SI	BÁSICA	
Categoría	aplica	+ R1	+ R1 + R2 + R3	op.exp.3	SI	SI	MEDIA	
Categoría	aplica	+ R1	+ R1 + R2	op.exp.4	SI	SI	BÁSICA (+R4)	7.6
Categoría	n.a.	aplica	+ R1	op.exp.5	SI	SI	MEDIA	
Categoría	aplica	+ R1 + R2	+ R1 + R2 + R3 + R4	op.exp.6	SI	SI	MEDIA	
Categoría	aplica	+ R1 + R2	+ R1 + R2 + R3	op.exp.7	SI	SI	ALTA	
T	aplica	+ R1 + R2 + R3 + R4	+ R1 + R2 + R3 + R4 + R5	op.exp.8	SI	SI	Medio	
Categoría	aplica	aplica	aplica	op.exp.9	SI	SI	BÁSICA	
Categoría	aplica	+ R1	+ R1	op.exp.10	SI	SI	BÁSICA	
Categoría	n.a.	aplica	aplica	op.ext.1	SI	SI	MEDIA	
Categoría	n.a.	aplica	aplica	op.ext.2	SI	SI	MEDIA	
Categoría	n.a.	n.a.	aplica	op.ext.3	SI	SI	ALTA (+R1+R2+R3)	7.7
Categoría	n.a.	aplica	+ R1	op.ext.4	SI	SI	ALTA	
Categoría	aplica	+ R1	+ R1 + R2	op.nub.1	SI	SI	ALTA	
D	n.a.	aplica	aplica	op.cont.1	SI	SI	Alto	
D	n.a.	n.a.	aplica	op.cont.2	SI	SI	Alto (+R1+R2)	7.8

Figura C.1: Declaración de Aplicabilidad del PCE-NIS2 (primera parte). Fuente: CCN-STIC-892, apartado 6.2 [3].

del 892), que para cada medida indica su aplicación a entidades esenciales e importantes y el nivel y los refuerzos exigidos. Se reproduce en las Figuras C.1 y C.2.

Codificación en el grafo

Cada celda de esa tabla se traduce en **propiedades de la relación** `CUMPLE_NIS2` (el "grafo gordo", Sección 4.7.2), recogidas en el fichero `reglas_pce_nis2.json`. La correspondencia entre la tabla del CCN y las propiedades del grafo es la siguiente:

- Aplicación en **mayúsculas** (BÁSICA/MEDIA/ALTA) → suelo *global*: `esencial_minima_global` / `importante_minima_global`.

Dimensiones								
Afectadas	CAT B	CAT M	CAT A	Medida	Importantes	Esenciales	Aplicación[1]	Ref
D	n.a.	n.a.	aplica	op.cont.3	SI	SI	Alto	
D	n.a.	n.a.	aplica	op.cont.4	SI	SI	Alto (+R1)	7.9
Categoría	aplica	+ R1	+ R1 + R2	op.mon.1	SI	SI	MEDIA	
Categoría	aplica	+ R1 + R2	+ R1 + R2	op.mon.2	SI	SI	ALTA	
Categoría	aplica	+ R1 + R2	+ R1 + R2 + R3 + R4 + R5 +R6	op.mon.3	SI *	SI	BÁSICA (+R2+R6)	7.10
Categoría	aplica	aplica	aplica	mp.if.1	SI	SI	MEDIA	
Categoría	aplica	aplica	aplica	mp.if.2	SI	SI	MEDIA	
Categoría	aplica	aplica	aplica	mp.if.3	SI	SI	MEDIA	
D	aplica	+ R1	+ R1	mp.if.4	SI	SI	Medio	
D	aplica	aplica	aplica	mp.if.5	SI	SI	Medio	
D	n.a.	aplica	aplica	mp.if.6	SI	SI	Medio	
Categoría	aplica	aplica	aplica	mp.if.7	SI	SI	MEDIA	
Categoría	n.a.	aplica	aplica	mp.per.1	SI *	SI	MEDIA (R1)	711
Categoría	aplica	+ R1	+ R1	mp.per.2	SI	SI	ALTA	
Categoría	aplica	aplica	aplica	mp.per.3	SI	SI	BÁSICA	
Categoría	aplica	aplica	aplica	mp.per.4	SI	SI	BÁSICA	
Categoría	aplica	+ R1	+ R1	mp.eq.1	SI	SI	BÁSICA	
A	n.a.	aplica	+ R1	mp.eq.2	SI	SI	Medio	
Categoría	aplica	aplica	+ R1 + R2	mp.eq.3	SI	SI	MEDIA (R1)	7.12
C	aplica	+ R1	+ R1	mp.eq.4	SI	SI	Bajo	
Categoría	aplica	aplica	aplica	mp.com.1	SI	SI	BÁSICA	
C	aplica	+ R1	+ R1 + R2 + R3	mp.com.2	SI	SI	ALTO (R4+R5)	7.13
I A	aplica	+ R1 + R2	+ R1 + R2 + R3 + R4	mp.com.3	SI	SI	Medio	
Categoría	n.a.	+ [R1 o R2 o R3]	+ [R2 o R3] + R4	mp.com.4	SI	SI	MEDIA	
C	n.a.	aplica	aplica	mp.si.1	SI	SI	Medio	
C I	n.a.	aplica	+ R1 + R2	mp.si.2	SI	SI	Alto	
Categoría	aplica	aplica	aplica	mp.si.3	NO	SI	BÁSICA	
Categoría	aplica	aplica	aplica	mp.si.4	NO	SI	BÁSICA	
C	aplica	+ R1	+ R1	mp.si.5	SI	SI	Bajo	
Categoría	n.a.	+ R1 + R2 + R3 + R4	+ R1 + R2 + R3 + R4	mp.sw.1	SI	SI	MEDIA	
Categoría	aplica	+ R1	+ R1	mp.sw.2	SI	SI	BÁSICA	
Categoría	aplica	aplica	aplica	mp.info.1	SI	SI (DNS)	BÁSICA	7.14
C	n.a.	aplica	aplica	mp.info.2	NO	NO	N/A	
I A	aplica	+ R1 + R2 + R3	+ R1 + R2 + R3 + R4	mp.info.3	NO	SI (PSC)	Alto	7.15
T	n.a.	n.a.	aplica	mp.info.4	NO	SI (PSC)	Alto	7.16
C	aplica	aplica	aplica	mp.info.5	SI	NO	Bajo	
D	aplica	+ R1	+ R1 + R2	mp.info.6	SI	SI	Alto	
Categoría	aplica	aplica	aplica	mp.s.1	SI	SI	BÁSICA	
Categoría	+ [R1 o R2]	+ [R1 o R2]	+ R2 + R3	mp.s.2	SI	SI	MEDIA	
Categoría	aplica	aplica	+ R1	mp.s.3	SI	SI	MEDIA	
D	n.a.	aplica	+ R1	mp.s.4	SI	SI* (DNS)	Alto	5.17

Detalles del criterio específico de aplicación de la medida en el apartado 7 de este documento.

Se considera que la medida, salvo excepciones, no será de aplicación al PCE-NIS2.

Figura C.2: Declaración de Aplicabilidad del PCE-NIS2 (segunda parte). Fuente: CCN-STIC-892, apartado 6.2 [3].

Listado C.1: Codificación de un control en `reglas_pce_nis2.json`.

```
"Op.acc.5": {
  "notas": "Mecanismo de autenticación (usuarios externos)",
  "esencial_minimo_nivel":        "Medio",
  "importante_minimo_nivel":      "Medio",
  "importante_excluye_refuerzos": ["r3"]
}
```

- Aplicación por **nivel dimensional** (Bajo/Medio/Alto) → suelo *dimensional*: `esencial_minimo_nivel` / `importante_minimo_nivel`.
- **`+R1, +R2...`** → refuerzos exigidos: `esencial_requiere_refuerzos` / `importante_requiere_refuerzos`.
- Refuerzos que **no** aplican a un tipo de entidad → `importante_excluye_refuerzos`.
- **n.a.** / medida no aplicable → `excluido_esenciales` / `excluido_importantes` / `excluido_pce_nis2`.
- Excepciones por **sector** (p. ej. proveedor DNS) → `sectores_especificos`.

Nota sobre el alcance. De estas familias, `sectores_especificos` (las excepciones por sector de actividad, como la inclusión forzosa de ciertas medidas para proveedores de DNS o prestadores de servicios de confianza) se **almacena en el grafo pero no la consume todavía el motor de inferencia**, pues requeriría incorporar el sector de la entidad al perfil de evaluación; queda como línea de trabajo futuro. El resto de familias (suelos, refuerzos y exclusiones) sí intervienen en el cálculo del Principio de Máximos. Las `condiciones_nis2` (notas de auditor y plazos de notificación) se trasladan como avisos informativos al evaluador, sin alterar qué requisitos resultan exigibles.

A modo de ejemplo, el control `Op.acc.5` (mecanismo de autenticación de usuarios externos) se codifica de la siguiente manera:

que se lee: la medida se exige a **nivel dimensional Medio** tanto a entidades esenciales como importantes y, a las **importantes**, se las **exime del refuerzo R3**, exactamente lo que recoge el apartado 7.3 del propio PCE-NIS2.

Criterios por medida

El detalle del criterio específico de aplicación de cada medida marcada en la Declaración de Aplicabilidad se encuentra en el **apartado 7** del PCE-NIS2 [3], al que se remite; no se reproduce aquí por su extensión.

Listado C.2: Estructura de `dora_mapeo_ens.json` (un artículo de DORA con su refuerzo).

```
{
  "id_dora": "DORA_7",
  "titulo": "Art. 9",
  "descripcion": "Art. 9 (3 b-d): controles que garanticen integridad y
                  exactitud de los datos, previniendo corrupción o pérdida...",
  "controles": [
    { "id_control": "Op.pl.2", "refuerzos_exigidos": ["r3"] }
  ]
}
```

C.2. DORA: *crosswalk* ENS–DORA (fuente secundaria)

A diferencia de la NIS2, DORA **no cuenta con un perfil de cumplimiento oficial** que lo proyecte sobre el ENS (Sección 3.2.1). Ante esa ausencia, la correspondencia ENS–DORA se ha tomado de una **fuente secundaria no oficial**: la capa de cumplimiento de **Proactivanet**, que relaciona los controles del ENS con DORA y NIS2 y permite explorar dicha correspondencia control a control [19]:

https://www.proactivanet.com/compliance-ens/

A partir de esa correspondencia se construyó el fichero `dora_mapeo_ens.json`, que **invierte la matriz**: agrupa los artículos de DORA junto con los Estándares Técnicos de Regulación (RTS) asociados y los conecta con los controles del ENS que los materializan, anotando los **refuerzos** que cada uno activa. Su estructura es:

En el grafo, cada entrada genera un nodo `ArticuloDORA` y una relación `CUMPLE_DORA` hacia el control del ENS, con la propiedad `requiere_refuerzos`. A diferencia de la NIS2, DORA no aporta suelos graduados por categoría, sino **refuerzos innegociables** que se activan cuando la entidad queda sujeta al Reglamento (Sección 3.3). El mapeo da lugar a **68 nodos** `ArticuloDORA` y **68 relaciones** `CUMPLE_DORA` (Tabla 4.1), centradas en el pilar de gestión del riesgo TIC (arts. 5, 6, 9, 10, 17 y 28; Sección 3.2.1).

Apéndice D

Flujo de extremo a extremo: diagrama de secuencia

Este anexo reproduce, mediante un diagrama de secuencia, el funcionamiento completo de la herramienta de extremo a extremo (Figura D.1). El Sistema Normativo (SN) se representa como **caja negra**: solo se muestran sus dos intercambios con el Coordinador (la entrega de los controles legislativos aplicables y la emisión del informe narrativo final), sin detallar su funcionamiento interno, ajeno al alcance de esta memoria. El recorrido refleja la orquestación implementada en `coordinator.py`: una conversación con el módulo del ST por cada control, anidada en el barrido de todos los controles.

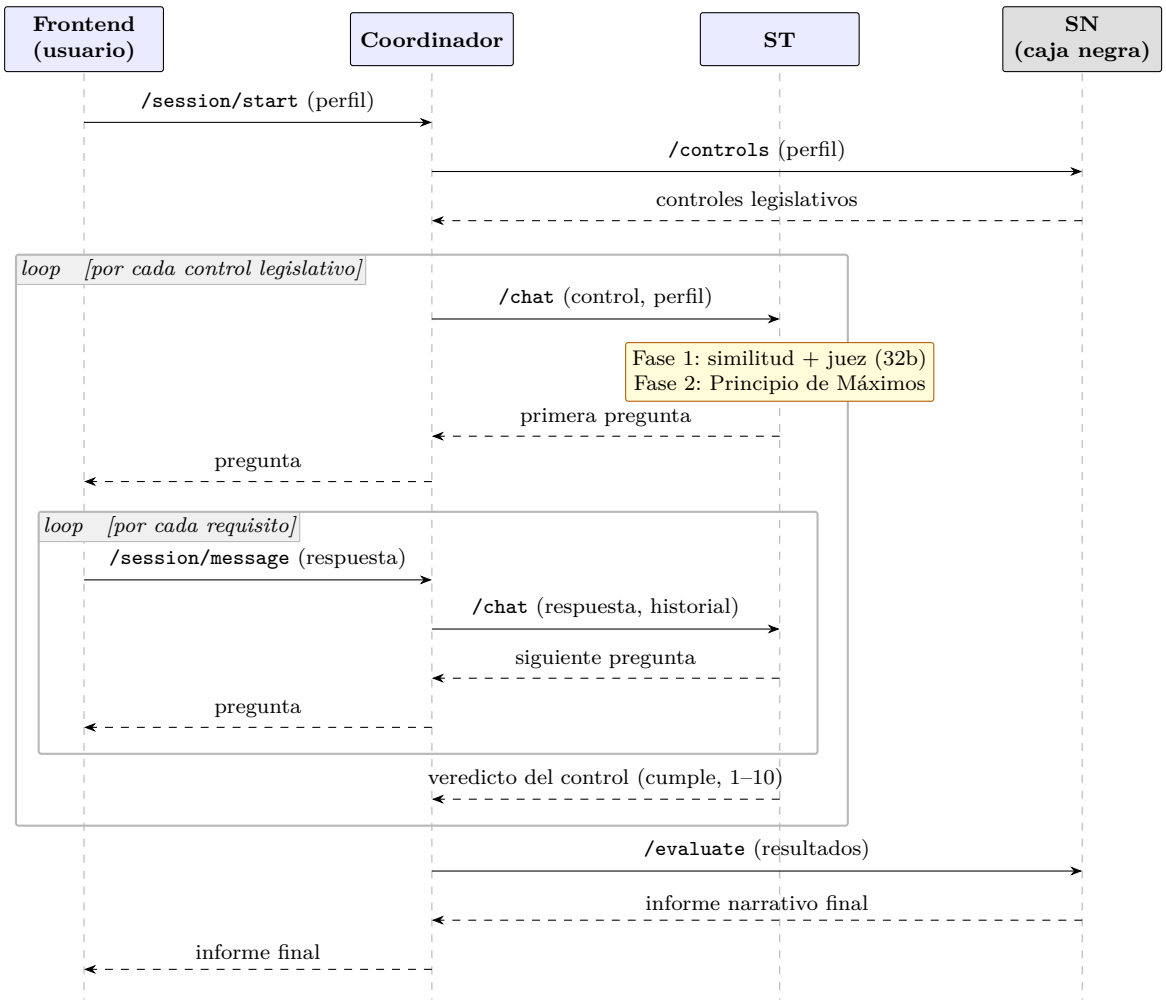

Figura D.1: Funcionamiento de extremo a extremo de la herramienta. El SN se trata como **caja negra** (solo sus dos intercambios con el Coordinador). La conversación con el módulo del ST por cada control, que internamente resuelve la Fase 1 (descubrimiento) y la Fase 2 (Principio de Máximos), se anida en el barrido de todos los controles legislativos.

Apéndice E

Funcionamiento interno del módulo del ST: diagrama de actividad

Mientras que el Anexo D muestra la herramienta de extremo a extremo, este detalla el **funcionamiento interno del módulo del Sistema Técnico** cuando recibe un control legislativo (Figura E.1): las dos fases del motor, el descubrimiento semántico (Fase 1) y el Principio de Máximos (Fase 2), la *caché* de veredictos que evita reevaluar un control ya resuelto, la auditoría conversacional requisito a requisito y la emisión del veredicto con su puntuación determinista.

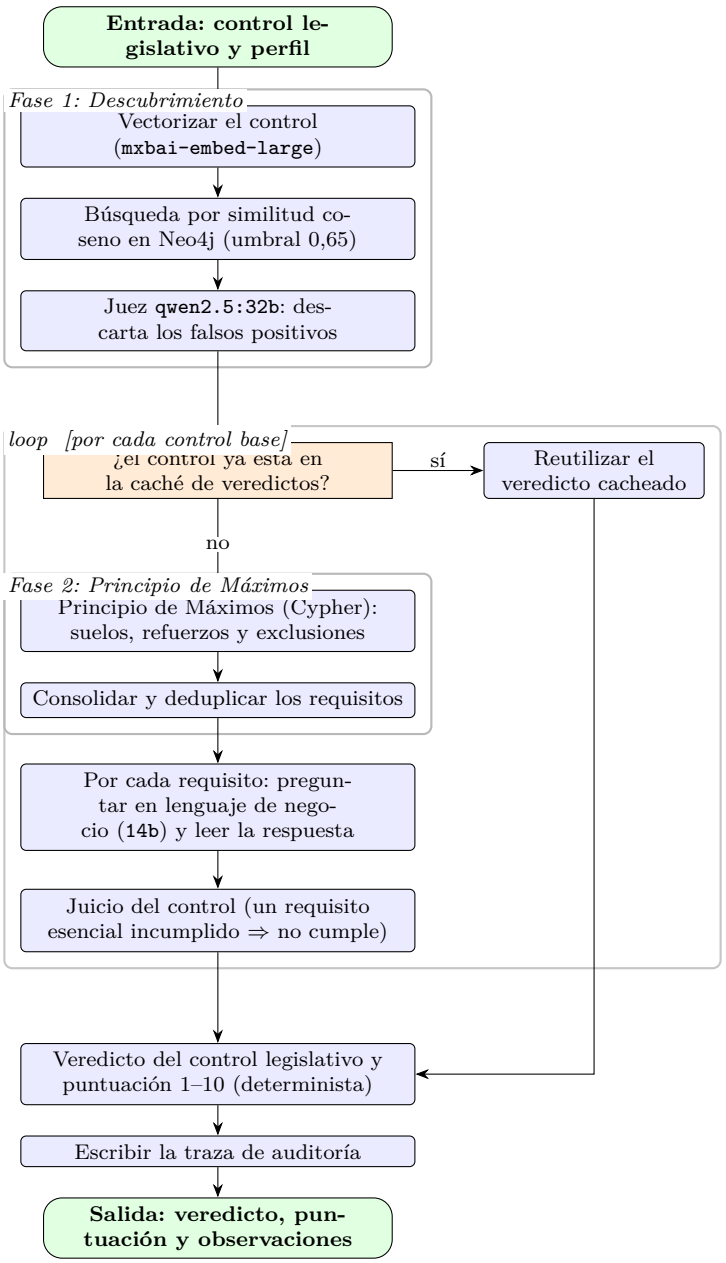

Figura E.1: Funcionamiento interno del módulo del ST. Tras descubrir los controles base pertinentes (Fase 1: similitud coseno y juez; si el juez falla, se conservan los de mayor similitud), el motor resuelve para cada uno el Principio de Máximos (Fase 2) y audita sus requisitos en conversación; un requisito *esencial* incumplido invalida el control. La *caché* de veredictos evita reevaluar un control ya resuelto en la sesión.